CHAPTER 7

Policy and Governance of Hybrid and Multi-cloud Infrastructure

Introduction

One way to think about policy is as an enforced objective. To be useful, a technology estate must be shaped to suit its purpose. It is constantly subjected to disruptive forces and requires not only that the initial design targets be met but constant tuning to ensure it continually aligns to its desired state.

Chief among the disruptors threatening an IT system's stability, security, and productivity for its intended use are the human beings that interact with it. Flaws are inherent in every physical and logical component of which an IT system is constructed, in the construction process itself, and the tendency to misuse or take expedient shortcuts will be present in those who ultimately use it.

R. Maxwell, *Azure Arc Systems Management*,
https://doi.org/10.1007/978-1-4842-9480-2_7

The innumerable theories as to why things devolve into chaos are the topic of some other book; this one is going to focus on how to fend it off, and the superhero saving the day is policy.

As noted in the close of the prior chapter, policy directly impacts security. This is true across the IT estate and from many perspectives. Operational policies can define how the infrastructure layer functions and when automated assure that best-practice security controls are implemented. Enterprise Content Management [ECM] policies can serve to protect business data and are a significant contributor to cost control efforts, as well as a tool to meet regulatory compliance standards. Policies can even serve as a training tool, as the guardrails they set make the organization's standards visible to users.

This chapter will look at several functional areas across the lifecycle of IT systems and their workloads and discuss their governance through the creation and application of policy. In each area, it will call out the opportunities to enhance security through correct policy management practices and will consider how Arc can extend Azure's policy engine across large hybrid IT estates.

Policy Scopes in Azure

The first obvious benefit is that the schema which you will hopefully have already applied to existing resources in Azure is now also available to Arc-enabled assets. An application running on Kubernetes in AWS is contained in a resource group you've designated for it in Microsoft Azure. If you have followed Microsoft's Cloud Foundations guidance for Azure, you will be familiar with the building blocks Microsoft provides to facilitate domain-driven architecture. The root of this architecture is always an Entra (formerly Azure Active Directory) tenant.[1] Underneath this tenant

[1] https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/landing-zone/design-area/azure-ad-define

is the first conceptual container of Management Groups. Imagine you are a large healthcare conglomerate with ownership of several health management organizations (HMOs), a few hospitals outside of those HMOs, pharmacy services, and a research facility in collaboration with a pharma company. With Management Groups, you can list out the factors for security, compliance, governance, resources, and so on that are common or disparate and arrange them under a Management Group that will be tuned to the correct requirements. For example, a research arm might need to integrate data feeds from other facilities or to manage the output of instrumentation used in research so that it can be analyzed. It might also have publishing requirements requiring submission workflows and access controls. That is a very different use case than a group focused on treating patients which might engage through remote visits or have heavy Enterprise Resource Planning [ERP] needs around clinic management. While commonalities like HIPAA regulations might show in the intersection of a Venn diagram describing those two use cases, there would also be large areas that do not overlap despite their similar industry, and placing them each in their own Management Group facilitates giving each group what it needs to be successful. Another example might be a multinational accounting firm. In many cases, the clients served are enormous and complex industries themselves, and placing the services relating to the client in a Management Group [MG] allows for a white glove approach to servicing that customer beginning with the fact that you can limit access to their MG to only personnel assigned to that client and their internal users. MGs are an extremely flat structure. Although the root MG has a one-to-one relationship with an Active Directory tenant beneath the root, the next level could consist of up to 10,000 sibling MGs. In contrast, the nesting ability for MGs is limited to six levels (excluding root). It's advisable to use MGs sparingly as very large buckets for key organizational divisions, as dividing by function is more appropriate at the subscription level. Policies or security controls applied at this level will cascade through the entire organization and cannot be reversed at lower levels in the

hierarchy, so only those few that are essential to security and operations but do not block normal activities lower in the hierarchy should be implemented at or near the root of the hierarchy. This is true not only for native Azure resources but also for those under Arc's control plane (with some discretionary selectivity using Resource Manager modes).

An Azure subscription is another confusing reference as it rather sounds like your entire purchase of cloud services but instead refers to a logical grouping below both the tenancy and the MG (if you don't create MGs, then all subscriptions automatically belong to the root MG). Continuing with the imaginary healthcare conglomerate, you could imagine two of the HMOs, each in their own MG and within that MG having multiple subscriptions to manage security and track costs for departmental functions such as patient records, appointments, ERP, marketing, consumer-facing applications, third-party integrations, and more. Subscriptions are discrete in terms of billing and policy and because they have a top limit on consumed resources close attention should be paid to subscription design for large workloads. Subscriptions cannot be in the same VNet, but can communicate over ExpressRoute or using virtual network peering (typically with a subscription dedicated to connectivity). Subscriptions can sometimes be moved to a new tenancy, a use case that can occur when spinning off a new business group, but only when created directly in your existing tenant vs. by a Cloud Solution Provider.

Within subscriptions live resource groups [RGs], which are the most common and familiar unit of organization within Azure. Resource groups are a way to organize groups of services, virtual networks, storage, and other artifacts that relate to one another in a logical or functional way (a dedicated RG is one of the first prerequisites to your trial deployment of Arc-enabled Kubernetes). Azure has physical data centers in regions around the globe, and since RGs are again a logical construct, they can contain resources from more than one region. They are a product lifecycle tool and tend to be applied to groups of things that are managed together whether that be a single application, a particular process, or a cost center.

If a business model evolves to place ownership of assets under another department, an RG can, with effort, be moved to a different subscription. Resources inside an RG can also generally be moved to another one if needed, but redeployment to the new RG would be more typical since moving an RG requires migration planning. For instance, a public IP address used by the RG cannot move and has to be disassociated, and then all of the artifacts depending on the IP address in the RG will have to be associated to a new public IP, which will have downstream effects anywhere that IP address is referenced. Overall, there are consequences in cost and complexity to creating more subscriptions than are truly needed, while RGs are a primary tool for organizing assets inside a subscription.

As we will dive into in the next chapter, policies present in the Resource Group, Subscription or Management Group are also applied to the foreign resources as if they were native by means of the Arc agent. Every server you enable (by installing a virtual machine extension) benefits not only from policy-based administration but also the ability to standardize machine configuration. The two features work hand in hand, for example, applying the policy `Deploy prerequisites to enable machine configuration policies on virtual machines` is a prerequisite to installing the virtual machine extension across groups of servers, as is also having a managed identity with the authority to execute the desired configuration changes. Even without the VM extension installed, you can apply policies to servers visible in the Arc dashboard as Azure objects (and at no cost), but the extension will be required for internal management of the VM and its operating system and will incur subscription charges. Since your servers are Arc enabled, they are now under the purview of Azure Resource Manager, and this means examining your server inventory, tagging, installation of extensions, and other common ARM tasks are free.[2] When

[2] https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/scenarios/hybrid/arc-enabled-servers/eslz-cost-governance#how-much-does-azure-arc-enabled-servers-cost

you utilize an installed extension to perform activities on an Arc-enabled server using its OS, you will be charged as the screenshot from the Azure Pricing Calculator[3] illustrates (Figure 7-1).

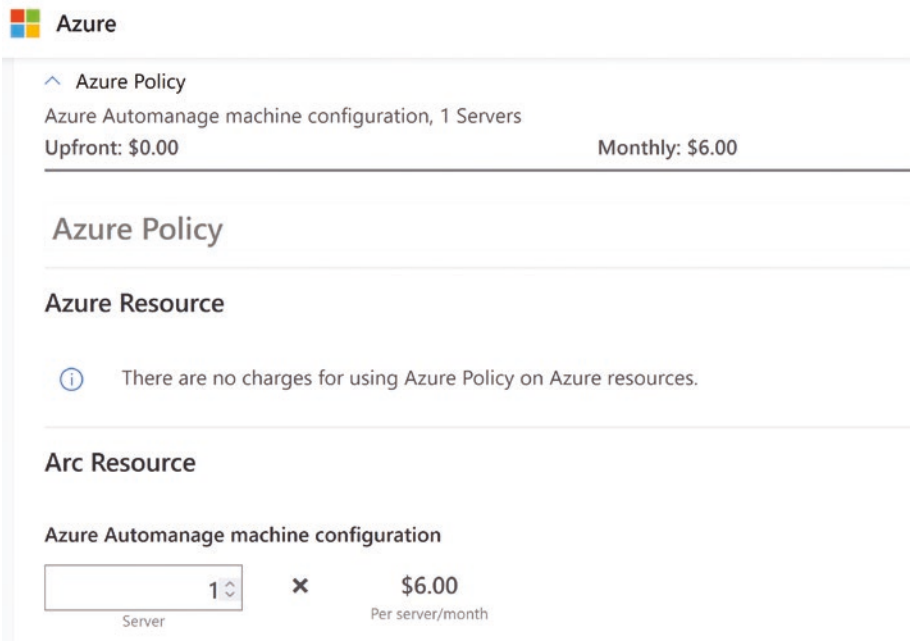

Figure 7-1. *Calculate cost of Azure Policy for Arc-enabled servers*

Azure Automanage Machine Configuration costs are offset by additional administration capabilities, for instance, you could assure that all of your servers, whether native to Azure or Arc enabled, use TLS.

[3] https://azure.microsoft.com/en-in/pricing/calculator/

Policy Baselines for Kubernetes

The ability to structure an environment through the implementation of policy is one of the most compelling advantages to a GitOps approach to Kubernetes deployments. You aren't limited to governing your GitOps pipeline at the time you first deploy clusters; rather, you can begin to apply policies anywhere in the lifecycle to begin reaping the benefit of having your installations automatically remediate deviation from policy.

There are built-in policy definitions for a number of scenarios[4] that Microsoft imagines will be useful in a large enterprise with a variety of implementations that address how rigid authorization standards should be or which git repository you will deploy from. Thus, you could have a cluster that allows for storing of secrets locally to facilitate rapid development cycles, but require certificates or the use of a private repository for high-value production assets. The Kubernetes API server is a primary attack vector, and protecting it correctly requires policies that provide the appropriate level of security to prevent inconsistencies and gaps.

> As Kubernetes is entirely API driven, controlling and limiting who can access the cluster and what actions they are allowed to perform is the first line of defense.
>
> —Kubernetes Documentation

The API server needs to be thoroughly hardened, thus simply applying RBAC will not be enough to assure its safety. An article on Akami's security blog[5] discussing proposed OWASP changes for 2023 points out

[4] https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/use-azure-policy-flux-2

[5] www.akamai.com/blog/security/proposed-new-changes-in-owasp-api-security

that injection attacks may soon be outranked as a top-tier risk in that authoritative list by Server-Side Request Forgery [SSRF] due to technologies like Kubernetes which pass URLs over APIs. Further basic controls include encryption and hardening network communications. The Shadowserver Foundation offers an interesting set of visualizations on their website showing unprotected Kubernetes API servers to which you can apply various geographic and time series filters, my favorite of which is the world view.[6] The count in some countries, including the United States, is in the hundreds of thousands. While some may not be high-value targets, the large number of exposed servers points to either a lack of awareness of the risks or unskilled teams overlooking basic security controls. Either way, it underscores the value of policy in creating immutable security controls.

Kubernetes' wide adoption has both surfaced its vulnerabilities and encouraged a set of best practices for protecting against them. Victims of crypto mining, ransomware, data breaches, and insider threats have shared hard lessons around the need for a consistent approach to repelling attackers. An article in the Sysdig blog discussing detection and multilayered policies to prevent crypto mining[7] attacks points out that the cost of cloud services to conduct a mining operation exceeds the return on the crypto itself manyfold, providing a powerful motivation for miners to operate on your cloud rather than buying their own (not to mention major cloud providers, including Microsoft as of December 2022,[8] updated their online services agreement to prohibit crypto mining on their platforms without specific permission).

Utilizing the Azure Policy engine is an advanced usage of the Azure ecosystem which is not limited to Arc-enabled resources; in fact, most of the items under discussion for securing Kubernetes clusters will apply

[6] https://dashboard.shadowserver.org/statistics/combined/map/comparison/

[7] https://sysdig.com/blog/detecting-cryptomining-attacks-in-the-wild/

[8] www.microsoft.com/licensing/terms/en-US/product/changes/MCA

equally to AKS and even the easy-entry Azure Container Apps service.[9] Most common Kubernetes distros which are CNCF certified can also benefit when they have the Arc extension installed.[10] It's important to have a holistic view of the elements which make policies effective, such as consistent tagging of resources and determining appropriate actions in response to policy violations. Policy is an important control plane element that must be combined with other facets such as GitOps deployments designed to restore a cluster to its desired state, logging of policy violations to discern whether they represent a security risk, and monitoring to assure policies enable profitable workstreams while protecting the organization's resources. It is this membership in the Azure ecosystem that makes the Azure Policy engine so powerful in comparison to stand-alone products which may have well-thought-out policy rulesets (such as the Falco Rules[11] used by Sysdig and others) but lack the automatic integration with the appropriate Azure services to respond to them. Microsoft's Zero Trust platform landed them an impressive top-tier spot in Forrester's Trust Platform Providers, Q3 2023 report,[12] which cites a centralized control plane and support of "diverse hybrid architectures" as selection criteria. Microsoft itself notes in touting its inclusion that it "delivers end-to-end cross-cloud, cross-platform security solutions, which integrate more than 50 different categories across security, compliance, identity, device management, and privacy, informed by more than 65 trillion threat signals … each day,"[13] a volume unmatched by most competitors across an also largely unmatched suite of platform capabilities.

---

[9] https://techcommunity.microsoft.com/t5/fasttrack-for-azure/azure-policy-for-azure-container-apps-yes-please/ba-p/3775200

[10] https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/validation-program#validated-distributions

[11] https://falco.org/docs/rules/basic-elements/

[12] https://reprints2.forrester.com/#/assets/2/108/RES179872/report

[13] www.microsoft.com/en-us/security/blog/2023/09/19/forrester-names-microsoft-a-leader-in-the-2023-zero-trust-platform-providers-wave-report/

Arc-enabled Kubernetes takes a proactive approach to policy implementation with a variety of built-in policies[14] intended to assure a uniform application of best practices for securing the clusters it deploys and manages. Security recommendations from the Kubernetes documentation highlight key surface areas to examine such controlling access to the API as previously mentioned and also to secure the network, kubelet endpoints, runtime users, and workloads, as well as cluster components, and we will explore how those surface areas are protected by correct policy implementation. While policy can and should be used to remediate existing installations,[15] it is important to view policy as an essential component of the initial configuration of Arc-enabled Kubernetes and data resources. Building large enterprise systems without initial policy controls is akin to manufacturing a powerful vehicle with no steering capability.

Policy files are written JSON notation and are simple to read and update. In the opening stanza of a policy, typical key-value pairs such as name and description are present, as well as the `policyType` key which defines whether the policy is `BuiltIn` (e.g., supplied by Azure) or `Custom`. This is also where the Resource Manager mode of either `all` or `indexed` is set for the policy. The default and typical value is to apply the policy to all resources. If you specify indexed, then the policy will be applied only to resources that support tag and location identifiers.[16] This could be useful for policies intended for a specific region or cost center but could fail to have the desired result if applied to unsupported resource types.

[14] https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/policy-reference

[15] https://learn.microsoft.com/en-us/azure/governance/policy/how-to/remediate-resources?tabs=azure-portal

[16] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/definition-structure#resource-manager-modes

As with any Azure service, the policy engine has capacity limits as to the number of policies, initiatives (sets of related policies), and lines per policy that are allowed.[17] The policy engine provides functions[18] derived from ARM template functions[19] to perform the necessary calculations to apply a policy, conditions for value matching, as well as logical operators (not, allof, and anyof). The ability to manipulate policy in this way is advantageous only if it's not used to cobble together workarounds for policy structure that was not well designed to begin with. Ideally, policy schema will be intrinsic to the architectural design of your systems from their genesis to enable their remaining compliant and consistent with their performance and security objectives. From your first Cloud Foundations baby step up onto the Azure platform to the installation of the Azure Policy extension,[20] you should know what you intend to govern with policy and how it will be defined in your deployments.

Selecting the correct scope and context for the application of policy should be viewed not just through the lens of a particular problem that a policy might prevent or remediate but also from a high-level observation of how the individual policy fits the policy schema for your organization as a whole. The Azure Policy engine is well organized for this approach, for instance, allowing policies with related objectives to be grouped into a policy initiative. It would be typical to have policy initiatives representing service areas or functional objectives (such as the governance of different

[17] https://learn.microsoft.com/en-us/azure/governance/policy/overview#maximum-count-of-azure-policy-objects

[18] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/definition-structure#policy-functions

[19] https://learn.microsoft.com/en-us/azure/azure-resource-manager/templates/template-functions

[20] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-for-kubernetes#install-azure-policy-extension-for-azure-arc-enabled-kubernetes

types of environments). It's important to remember that if a resource is subject to an initiative, it must pass the muster for *all* of the policy definitions contained within that initiative, so if a policy should only be applied under atypical conditions, it is better not to include it in an initiative.

The outcome of the application of a particular policy is simply referred to as its effect. Effects[21] are not random, but rather specify what the policy will accomplish such as to modify an artifact, deny access, audit only, or audit and deploy. If multiple effects are implemented in the same policy, their order of precedence must be considered; otherwise, they may not work as intended.[22]

Manual effects are a new type of effect to allow interactive response to compliance policies. They[23] are particularly suitable for meeting compliance regulations in financial and healthcare fields that require a human being to confirm some action has been performed. The JSON structure of a manual effect specifies the objective and sets its initial status to "unknown." The response to the policy is provided through a separate resource, the Azure Policy attestation structure,[24] which is basically a mini workflow engine that captures metadata regarding the task owner and details of the response, including its updated compliance status, commentary, and evidentiary links to supporting documentation.

[21] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects

[22] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects#order-of-evaluation

[23] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/effects#manual *and* https://learn.microsoft.com/en-us/azure/governance/policy/concepts/attestation-structure

[24] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/attestation-structure

Figure 7-2. *Sometimes, policies require a human touch*

Microsoft defines Policy as Code [PaC[25]] as the intersection of IaC and DevOps, but then takes it a step and an acronym further with EPaC, or Enterprise Platform as Code,[26] which is designed for organizations that must maintain extensive policy libraries and includes approaches for integrating policy with Azure Landing Zones deployments as part of the Cloud Adoption Framework [CAF].

In the Azure portal's Policy ➤ Definitions section, there are search tools for the library of policies and the ability to sort by the platform area you are targeting, whether you are looking for a policy or an initiative, and to filter by keyword (which brings up just under 20 built-in policies containing the phrase "Arc-enabled"). While Microsoft has at the time of this writing disabled the ability to directly export a policy to GitHub, the JSON policy

[25] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-as-code

[26] https://azure.github.io/enterprise-azure-policy-as-code/

definition is able to be pasted into an editor and subsequently committed to the repository in which you will manage the PaC. In whatever manner you accomplish it, this will be a necessary step as every policy will need customization to run under the correct service principals and target the actual resources existing in your organization. People with command-line expertise are likely to appreciate the ability to search for policy definitions using a PowerShell cmdlet piped to a where clause and the extensive policy descriptions provided in the shell, allowing for an entirely scripted approach to policy selection and implementation. Policy can also be described and implemented using Terraform or REST calls – all part of the ease and usability of living in an ARM world.

With a Policy as Code[27] approach, testing or validating policies is as important as it would be in preparing the production release of any other type of code. Policies can be designed to execute enforcement actions; therefore, you must assure that your policy doesn't return false positives, and new policies should have their enforcement mode set to disabled until their veraciousness is established.

Network Policies

If you could only choose one target for the policy engine, the network would be the bullseye for which to aim. Motor through a slew of recent published breaches and the network as the communications infrastructure providing required functionality for your workloads is also the roadway used by those with criminal intent. As the seventh item on the 2022 OWASP Top 10 list of Kubernetes security vulnerabilities highlights, with Kubernetes flat networking structure, "when no additional controls are in place any workload can communicate to another without constraint.

[27] https://learn.microsoft.com/en-us/azure/governance/policy/concepts/policy-as-code

Attackers who exploit a running workload can leverage this default behavior to probe the internal network, traverse to other running containers, or invoke private APIs." What does this mean in practice?

Port scanners use a brute-force approach to find either open ports or ports of specific interest because they belong to an asset that has a known path for intrusion. One common example would be a manually installed instance of PostgreSQL that utilizes trust authentication to allow database access for any user able to connect (PaaS offerings of PostgreSQL from Azure, AWS, Google, and others do not allow this setting). Over the past couple of years, this has given the old-school cryptomining malware Kinsing a new playground in the form of Kubernetes clusters. In terms of policy, if you run large Kubernetes installations, you should be assuring that none of your PostgreSQL servers allow trust authentication.

Older versions of Oracle WebLogic, a popular Java application host, are also vulnerable to Kinsing, and Microsoft security researchers note attackers start out searching for WebLogic's default port.[28] Although minimizing the amount of protection changing SQL Server's default port actually provides, since TCP/IP connections allow port scanning as a matter of course, Microsoft to this day provides guidance on how to change it.[29] The problem with ports is not limited to those known to be vulnerable, just having too many open can provide potential routes to your systems. Does the creation of a VNet in your org automatically include a public IP? Why? Turn off or limit that capability via policy.

With a chief advantage of Arc being access to the Microsoft ecosystem of products, the aforementioned built-in policy requiring it be turned on is particularly applicable in the face of known threats like Kinsing, the attack

[28] http://techcommunity.microsoft.com/t5/microsoft-defender-for-cloud/initial-access-techniques-in-kubernetes-environments-used-by/ba-p/3697975

[29] https://learn.microsoft.com/en-us/sql/database-engine/configure-windows/configure-a-server-to-listen-on-a-specific-tcp-port?view=sql-server-ver16

vectors for which are already shielded when using Defender. But further than that, whomever you rely upon for additional layers of security has to have a laser focus on the ever-changing threat landscape so that gaps can be filled the moment they are anticipated. Security is very much an activity on which having partnerships matters, so that your company doesn't have to review each CVE and response solo. Manual defense strategies are simply not an option in the age of automated threats.

While Kubernetes natively provides protective ingress and egress policies that can, for example, restrict incoming traffic to a known range of IP addresses or forbid outgoing calls, a significant advantage to Azure Arc is the ability to use Azure Private Link[30] which provides a connection to the resources in your cluster that does not travel over the public Internet, thus greatly reducing the exposure of your traffic to both human and automated threats. Further, Private Link offers control over who connects to services and the scope of services available limiting the potential for data leakage.

Policies governing traffic between pods are a baseline safety configuration to prevent infection or intrusion spreading throughout your cluster by taking advantage of Kubernetes natively open structure. It's important to remember that pod policies are aggregate, meaning that if you have one policy stating that ingress traffic is allowed to pods with the label `customer-address-db` from pods labeled `web-front-end1` and a second policy allowing ingress traffic from `web-request-api`, *both* ingress paths will be allowed, not just the API request. Thus, security again becomes a matter of both appropriate design and testing to assure that traffic follows the path you intend. Likewise, you need policies on both ends of the pipe. If you have no egress policy permitting the pod hosting the API to connect to the database, the communication will fail.

[30] https://learn.microsoft.com/en-us/azure/azure-arc/servers/private-link-security

Policies Governing Containers

Containers are a core policy objective in securing and managing your Kubernetes installation. They are in fact the treasure you guard with Microsoft Defender for Arc-enabled Kubernetes installations. Key considerations in protecting your containers are their source repository, preventing privilege escalation via their operating systems and controlling network access.

A new attack hijacking RBAC controls was caught in a honeypot set up by Aqua[31] (a firm that sells a cloud-native security platform specializing in protections against supply chain and Kubernetes attacks). Unless your clusters are antique or were set up poorly to begin with and grant anonymous access requests to a cluster admin role (or malformed third-party tools have done so), you are likely safe from this particular intrusion,[32] but the two things that are interesting about it are that it escapes typical monitoring by creating a service account with a name mimicking the kube-controller so that role binding that accounts to cluster admin privileges doesn't necessarily raise alarms, and when the hack gains admin privilege and pulls the containers contaminated with the cryptomining software, they are sourced from a public Docker registry. An image pull policy[33] restricting which image registries can be used to populate your clusters is a vital part of both DevOps and GitOps pipelines.

[31] https://blog.aquasec.com/leveraging-kubernetes-rbac-to-backdoor-clusters

[32] https://access.redhat.com/articles/7009182

[33] https://portal.azure.com/#view/Microsoft_Azure_Policy/PolicyDetailBlade/definitionId/%2Fproviders%2FMicrosoft.Authorization%2FpolicyDefinitions%2F50c83470-d2f0-4dda-a716-1938a4825f62

Public registries should *not* be used in the enterprise, even for development, since they introduce the risk of unmanaged images into your ecosystem. While you can build a private registry, they are readily available from all major cloud vendors from Red Hat and AWS to Microsoft and Docker and can be configured to integrate with your own RBAC systems.

For a private registry to serve its purpose, what is allowed to be pushed into it and who is allowed to place assets there are the factors that will determine its effectiveness; thus, images should conform to security and performance specifications as well as be digitally signed by contributors. Even with safeguards in place, containers within your registry should also be regularly scanned for vulnerabilities[34] either with Microsoft Defender's Qualys scanner if you are using the Azure Registry or any one of a number of quality tools from vendors like Aqua. An Azure Container Registry policy to assure the scanner is run before a container can be digitally signed and uploaded should be enabled when using Defender.[35] Policy is not the only avenue to assure containers are scanned as many scanning tools integrate with DevOps pipelines so that running a scan is part of a build. On GitHub, a container vulnerability scan is available as an action. A policy engine is not governance, but a governance implementation tool, so you are free to implement the governing standards of your organization in the most expedient way. The advantage of policy is that it is declarative, auditable, can be versioned, and is very easy to reference from a governance document in order to demonstrate compliance. In the case of Azure, built-in policy templates exist for every important service providing a baseline

[34] https://learn.microsoft.com/en-us/azure/container-instances/container-instances-image-security#monitor-and-scan-container-images

[35] https://portal.azure.com/#view/Microsoft_Azure_Policy/PolicyDetailBlade/definitionId/%2Fproviders%2FMicrosoft.Authorization%2FpolicyDefinitions%2F090c7b07-b4ed-4561-ad20-e9075f3ccaff

for operational efficiency and security so that you can start in a good spot and grow from there as you develop more sophisticated policies aligned to your business model.

Microsoft advises collocating your registry as close as possible to the containers will be deployed to cut down on egress fees, which can be substantial. Imagine you have thousands of images whose base layer must be updated (perhaps due to an upgraded release or newly identified vulnerability), and you have automated a rebuild and redeployment of all images built on that base. A retailer for whom I worked specifically negotiated reduced egress fees when their cloud provider pushed a mandatory upgrade to its most popular Linux base image. When Cloudflare recently began offering S3-compatible storage with no egress fees,[36] one company actually saw that as an opportunity to build a new container registry,[37] and organizations who run their own registries are likely paying attention.

When securing your enterprise against threats, you could understandably go into CVE overload at the relentless onslaught of new threats. If you focus instead on ranking the criticality of your own assets in terms of what you want to protect and also realize that many "new" CVEs are really recycling the same techniques, you will be able to design a hardened infrastructure and will be able to limit damage or prevent it altogether (however, if you do want to feed the worry monster, visit the official Kubernetes CVE feed[38] which refreshes daily).

Sometimes, it's a combination of Kubernetes design and its platform that causes issues. Palo Alto Networks published a succinct summary[39] of how Kubernetes native design lent itself to "node-to-admin privilege

[36] https://tinyurl.com/cakpbyk2

[37] www.wired.com/story/container-registry-security-chainguard/

[38] https://kubernetes.io/docs/reference/issues-security/official-cve-feed/

[39] https://unit42.paloaltonetworks.com/kubernetes-privilege-escalation/#acc5816d-8466-4eaa-8349-e7855919a873

escalation" by replacing DaemonSets[40] (a type of administrative controller that can facilitate activities like logging and monitoring on a node and ensures the node is running a pod) with an evil version that operated like a cuckoo bird chick, knocking all the other nodes out of the cluster nest by tainting them as `NoSchedule` and then using a `NoExecute` taint on the node containing the pods it wishes to consume, forcing the cluster to recreate them on the only bird left in the nest – the imposter. The article is retrospective and so was able to share how both AKS and EKS now prevent this scenario using validating admission controllers,[41] and understanding their usage is also essential to securing the clusters you deploy that are not part of a managed service like AKS.

Much of what has been discussed in terms of protecting Kubernetes deployments via policy is simplified by using Defender for Containers,[42] which will suggest where the policy extension can be added to Arc-enabled servers and then continuously monitor policy compliance wherever it is enabled. This is active monitoring in that admission requests to the Kubernetes API must pass a Gatekeeper webhook that checks whether policy constraints are met before the request can proceed. These policy checks are in addition to the set of best-practice rules originating in Defender itself and the resulting suggestions it will offer in terms of policy

[40] https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/

[41] https://kubernetes.io/docs/reference/access-authn-authz/admission-controllers/#validatingadmissionwebhook

[42] https://techcommunity.microsoft.com/t5/microsoft-defender-for-cloud/leveraging-defender-for-containers-to-simplify-policy-management/ba-p/3755757

and general security management. Defender for Containers[43] is a core value-added provided by Arc-enabled Kubernetes and can be enabled for containers in all three commercial clouds.

When discussing protective policies for containers, it's noteworthy that many of the built-in policy definitions in Azure aim to protect against the ability to run any container's operating system in administrative mode for Windows or as Linux root. A CrowdStrike blog[44] discussing the mechanics of a particular CVE makes this elucidating comment, "Linux kernel exploits are an alternative method to escape container environments to the host in case no mistakes in the container configuration were made. They can be used because containers share the host's kernel and therefore its vulnerabilities, regardless of the Linux distribution the container is based on." Thus, after you have correctly set up your GitOps pipelines, are protecting your supply chain, and have remediating policies to assure your configuration remains pristine, you still must consider hardening the container OS. Cloud vendors are absolutely paying attention to these risks, which threaten not only your workloads but theirs as well. As Google explains in an update to their Vulnerability Reward Program (which has been paying bounties relating to GKE since 2020), they wish to "support researchers evaluating the security of Google Kubernetes Engine (GKE) and the underlying Linux kernel. As the Linux kernel is a key component not just for Google, but for the Internet…"; they have redirected the program toward hardening the kernel itself.

Microsoft has devoted similar energy for a number of years, and the result is the release of Azure Linux, a hardened distribution that was another great fruitage of the Avanade and Microsoft collaboration on

[43] https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-containers-introduction?toc=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2Fazure%2Faks%2Ftoc.json&bc=https%3A%2F%2Flearn.microsoft.com%2Fen-us%2Fazure%2Fbread%2Ftoc.json

[44] www.crowdstrike.com/blog/exploiting-cve-2021-3490-for-container-escapes/

behalf of mutual customers. Mike DeLuca explains that Avanade put in special requests in terms of security that Microsoft met and exceeded, asking "we want to know what's open and what's closed" in a new hardened distro that could protect the massive asset base running on Azure. Commercial releases each have their vagaries, for instance, the June 2023 Ubuntu Linux kernel update[45] lists fixes to their distribution that impacted all three major cloud providers. Becoming the responsible party for the Linux distro underlying major services like AKS and Edge gives Microsoft the opportunity to resolve essential security and performance obstacles with a direct line of site into a core platform OS. DeLuca expressed satisfaction with the outcome, commenting, "It's their own Microsoft opinionated distribution – *secure*, everything turned off until you turn it on."

CBL-Mariner Linux was a fresh build from the ground up (with small exceptions related to "borrowing specs" from distros like Fedora and CentOS) rather than a customization of an existing Linux distro. This approach avoids derivative patching cycles and shared vulnerabilities that could have resulted from choosing to modify even an admirable published Linux distribution. While a discussion of the history of Azure Linux might seem out of place in a policy discussion, it is relevant since hardening the OS fills a gap that *both* monitoring and policy have trouble reaching. Kernel exploits often take extreme measures to make their operations look normal until their assault on vulnerable systems is complete, and by the time such exploits are underway, policy gates have already failed or were insufficient. Microsoft starts its Azure Linux distro with a compact core of around 400MB with, as of Microsoft Build 2023,[46] about 300 additional packages available that have been thoroughly tested for compatibility and safety. The distinction between the CBL-Mariner Linux project and Azure

[45] https://ubuntu.com/security/notices/USN-6171-1

[46] https://build.microsoft.com/en-US/sessions/e84dd80a-f3bb-4d3d-978e-ffd811e3bfe1?source=sessions

Linux is that Azure Linux distro is the commercially supported version and is already in use internally at Microsoft for products such as AKS, Azure Stack Edge, Azure Arc, and more.

Data Policies

If there was ever an argument for well-crafted policies, Azure's 2023 ten-hour plus outage of SQL databases in their South Brazil DC[47] provides it. While upgrading legacy packages in the Azure supply chain, a "typo" caused the unintended deletion of entire database servers, triggered by a condition having to do with the age of database snapshots. Arguably, the role with access to manage snapshots should not also be able to delete the database server itself, which could indicate one missed opportunity for the application of policy. To restore all 17 production databases wiped out by the accidental deletion of their host, a new SQL Server had to be set up and the databases restored from backup. At this point, a second major delay was caused by inconsistent DR approaches, with some of the databases having been "… created before Geo-zone-redundant backup was available." This meant when using Geo-zone-redundant restoration for the legacy databases, additional hours waiting for the database to be copied into a secondary zone[48] greatly extended the recovery time. This scenario appears to beg for a GitOps policy that would have audited the DR strategy for each database and offered a path to remediate any legacy databases not using Geo-zone-redundant backup. This is especially true since long-standing database instances may have not have been initially created via a GitOps pipeline, but perhaps by an administrator using T-SQL in which case even if policies existed, they would not have been applied. Even as the databases were restored and brought back online, an overwhelming

[47] https://status.dev.azure.com/_event/392143683/post-mortem

[48] https://learn.microsoft.com/en-us/azure/storage/common/redundancy-migration?source=recommendations&tabs=portal

number of customer requests forced Microsoft to "[block] all … to allow all web servers to warm-up and successfully enter the load balancer," thus extending the outage even further, severely impacting business workloads running in that location.

None of the takeaways Microsoft listed in their mea culpa on the outage were innovative or new strategies, all could potentially have been implemented in the normal course of operations. The real object lesson here is not just preparation but having the sort of creative imagination that would fuel a great Hollywood disaster flick. The more adverse scenarios affecting your systems your most vociferous Murphy's Law advocates can come up with, the better you will be able to design a policy strategy to protect them. The principles of RBAC and least privilege are well known, but discovering every opportunity to apply them will require a thorough understanding of your systems and their workloads. While Azure Arc gives you the reach to apply policies to your systems running anywhere, what to apply will be a primary concern.

Arc in and of itself is not auto-magical. For instance, installing the Arc agent on a group of SQL Servers does not enable backup – in fact, because there are potentially already backup routines in place on the servers being onboarded, automated backup is explicitly *disabled*, and you must specifically enable it in order to benefit from Arc's consolidated management of routine operations. If your SQL Servers are part of an Always On availability group, you will only be able to automate backup on the primary replica, and servers that have more than one database instance require you to instantiate automated backup on each individual instance; however, the SQL Server extension that facilitates these management tasks only has to be installed once per server, and that task can be accomplished by a `DeployIfNotExists` policy that requires its installation on every Arc-enabled machine running SQL Server.

Once the SQL extension is installed, the first built-in policy you may want to execute against SQL Servers running anywhere under Arc may be to enable a SQL best practice assessment[49] for those servers (keeping in mind that running the assessment will subtract from the server's ability to run its customary workload), which applies several hundred rules[50] to assess a wide range of configurations from indexes and backup compression to whether you have properly set up OLTP resource pools on your Windows-hosted SQL Servers. The data collected is saved to a Log Analytics workspace where you can perform further analysis as desired and determine candidates for remediation steps that can be performed as part of the policy execution or as separate administration tasks.

Policy is governance applied, and nowhere is this more credible than in the management of data platforms and data. Which data, if compromised, could threaten your business model? Which data offers new or underutilized revenue opportunities? What are the operational blockers to accessing the data you own and extracting full value from it? How close are you to real-time processing of data if you are in an industry where there is a constant flow of new information? Policy is a subset of overall systems architecture that must be continuously evaluated in the light of current business conditions and objectives for as long as the business or organization continues to operate an information technology platform.

Remediating Existing Resources via Policy

One of the most powerful aspects of Azure's policy engine is that it is not limited to reporting policy exceptions, but has the ability to remediate them. Keeping in mind the risks of system changes in general, a well-vetted policy is an excellent foundation for automated systems administration.

[49] https://learn.microsoft.com/en-us/sql/sql-server/azure-arc/assess?view=sql-server-ver16

[50] https://github.com/microsoft/sql-server-samples/blob/master/samples/manage/sql-assessment-api/DefaultRuleset.csv

Policy remediation can occur at several junctures. For instance, you might have a policy feedback loop (methods of approaching this are discussed in the next chapter) that informs of new threats and vulnerabilities requiring either audits or changes to various policies, essentially reactive policy implementation procedures. You can also set up proactive policies that automatically remediate any out of compliance resources to assure that key policies remain in compliance. Additionally, policy remediation can flow from the normal architecture of system lifecycles which may indicate some policies can be deprecated and also suggest policy additions or refinements that can require you to remediate an existing policy.

When crafting a remediation task triggered by a policy exception, it's important to remember that the identity executing the remediation must have the appropriate role privileges to perform whatever that particular remediation action consists of. There is a slight advantage to creating a managed identity through the Azure portal, as permissions to the task contextual to the identity creation are automatically granted,[51] but you may wish to grant them specifically in accordance with how you have designed your RBAC controls.

GitOps and Policy Development for Kubernetes

The GitOps workflow for Arc-enabled Kubernetes is an industry standard inner loop pattern,[52] meaning the first iteration of coding and testing is performed on the developer's workstation before it moves to the outer loop of the cloud ecosystem where it can be trialed on development servers that mimic production, undergo various further refinements

[51] https://learn.microsoft.com/en-us/azure/governance/policy/how-to/remediate-resources?tabs=azure-portal#grant-permissions-to-the-managed-identity-through-defined-roles

[52] https://developers.redhat.com/articles/2022/12/12/kubernetes-native-inner-loop-development-quarkus#container_based_inner_loop_solutions

such as integration or security testing, and then finally be published to production servers. To achieve consistency, security, and compliance with development patterns and practices, some companies (Google, for example[53]) sometimes use a cloud workstation as the developer machine. Microsoft's DevSpace framework approach to the inner loop pattern[54] allows for coding on the developer's local machine but provides hooks into the cloud for remote debugging and also automates container provisioning so that the developer can sink into code authoring with adjacent deployment tasks abstracted away. When ready, the framework then provides a transition to the outer loop.

Once the development pipeline is set up, your devs are ready to author policies, as well as implement and test them programmatically.[55] While a good workflow as recommended in the docs will contribute to success as it would in any other development project, the most important thing to call out here is that *it is* a major development project and as such should have a solid architectural runway, business oversight that includes program management, and an everlasting CI/CD pipeline that assures the policies used to govern your IT assets remain equal to constantly changing technology stacks, security threats, and business objectives. Effective policy administration is part of an overall architecture for your systems and should never be viewed as "set it and forget it" because like an aging dam it will develop cracks and fail if neglected with perhaps disastrous consequences.

Illustrating the value of a continuous integration approach to policy, in October of 2022 Microsoft reported a vulnerability affecting Arc-enabled Kubernetes clusters and potentially any Azure Stack Edge or

[53] https://codelabs.developers.google.com/innerloop-dev-cloud-workstations-nodejs#0

[54] https://learn.microsoft.com/en-us/azure/azure-arc/kubernetes/conceptual-inner-loop-gitops

[55] https://learn.microsoft.com/en-us/azure/governance/policy/how-to/programmatically-create

AKS installations connected via Arc.[56] If an attacker were able to sniff the DNS endpoint for a cluster, they could potentially take administrative control of the entire cluster. Both Azure Arc-enabled Kubernetes agents and Azure Stack Edge require version upgrades to fortify clusters against this attack vector. Although it's possible to set the agent to poll for updates hourly and automatically upgrade using Helm, situations like this provide an opportunity to use the policy engine advantageously when you have large Kubernetes farms and must assure there are no outliers that remain exposed.

If you consider that best practice for Test-Driven Development [TDD] is to write tests prior to writing code, that principle translates well to policy development in that anticipating effect of policy not only as to content but also in terms of the scope it will best be applied to and any other consideration that will assure the policy is effective and not easily circumvented is important.

[56] https://msrc.microsoft.com/update-guide/en-US/vulnerability/CVE-2022-37968

Policy for IT Consumers

Figure 7-3. *Policies should enable worker safety and productivity*

At the time of this writing, a worldwide data breach caused by a SQL injection vulnerability in a file transfer utility, MOVEit,[57] which improves upon antiquated FTP protocols with the intention of ensuring the safety of data like bank records and personal health information by utilizing encrypted transit protocols is underway. The CIOp ransomware group believed to be responsible for the breach may have been executing trial runs for this large-scale attack years in advance according to SecurityWeek.[58] The Federal News Network[59] lists the US Energy Department and Veterans Administration among those targeted and also

[57] www.progress.com/security/moveit-transfer-and-moveit-cloud-vulnerability

[58] www.securityweek.com/evidence-suggests-ransomware-group-knew-about-moveit-zero-day-since-2021/

[59] https://federalnewsnetwork.com/cybersecurity/2023/06/energy-department-among-several-federal-agencies-hit-by-moveit-breach/

notes that universities and private businesses including Shell are believed to be affected. The State of Oregon's Department of Motor Vehicles couldn't identify exactly which assets were stolen in the MOVEit hack and so informed everyone who has a driver's license in that state to consider their identity at risk,[60] while the UK payroll provider Zellis use of the software has impacted many of their customers.[61]

While the MOVEit breaches are *not* impacting Kubernetes clusters directly, Mandiant[62] and other security research firms do state that Azure Storage is impacted with Baker Donelson claiming, "The vulnerability allows for unauthorized data access and control, including access to Azure Storage Blob credentials, providing the attackers with the means to steal data directly from a victim's Azure Blob Storage containers. This situation is particularly alarming because, with a copy of the database, threat actors can continuously attempt to access even encrypted data, posing a severe threat to organizational security."[63]

I mention them here because they illustrate the risks inherent in a mixed stack that includes third-party tools, custom applications that may be affected by supply chain vulnerabilities, internal and external connectivity, and the oftentimes sensitive data on which all of these rely. Additionally, the greater the potential damage from an insecure implementation in any of these areas, the louder the clamor to hold someone accountable will be. In March of 2023, the current administration

[60] www.oregonlive.com/commuting/2023/06/massive-hack-of-oregon-dmv-system-puts-estimated-35-million-driver-license-and-id-card-info-at-risk-officials-say.html

[61] https://techmonitor.ai/technology/cybersecurity/zellis-cyberattack-british-airways-boots-bbc

[62] www.mandiant.com/resources/blog/zero-day-moveit-data-theft

[63] www.bakerdonelson.com/moveit-transfer-zero-day-vulnerability-what-companies-need-to-know

of the United States published a National Cybersecurity Strategy[64] which commented in part, "We must begin to shift liability onto those entities that fail to take reasonable precautions to secure their software" and also notes that "end-users… often bear the consequences of insecure software," which will be painfully true for large groups of people in this particular case.

As organizations provide the tools, their users need to effectively do business the pace at which change happens is incredibly rapid. Arc particularly shines in the area of accidental hybrid IT estates, where the need for uniformity of governance was never more apparent than it is today. As companies move toward the cloud, consolidate and acquire new lines of business, or even repatriate some assets to privately owned DCs, the need for effective policy to manage disparate resources is readily apparent. Yet if you took only the case of acquisitions, a misapplication of needed controls could leave users feeling unsupported or worse resentful of the changes being made to familiar work routines amid their companies' transition to new ownership.

Policy controls become more palatable when they are framed as an opportunity to contribute meaningfully to the security of the company to which the user subject to the policy is devoting their efforts. Policy guardrails are not intended to prohibit a user from reaching what they need in an efficient manner. To craft policy that enables productivity, the policy designer has to understand the user's workflow while remaining aware of the greater risks posed by potential gaps in security or process. In the case where a policy needs further tuning to meet these objectives, IT must not be reactive – disabling an important policy due to a user complaint – but rather incorporate user acceptance testing into the policy's development lifecycle in a process of continuous improvement.

[64] www.whitehouse.gov/wp-content/uploads/2023/03/National-Cybersecurity-Strategy-2023.pdf

If you think about the number and type of policies impacting end users such as eDiscovery, document retention and classification, permission to assets associated with their own role, and more, many of them will live within content management, financial, and document processing systems that already have defined governance controls for working with typical business process flows. But if you start to delve into the infrastructure that supports many of the same tools – the file transfer product mentioned in the introduction of this heading being an excellent example – then policy reverts back to being a platform governance issue most of the time. Whatever useful tool is facing the user, if there is a data store behind it or network traffic exiting the boundaries of your organization, it will need to be governed and secured, with enterprise-wide application of policy being a core tool to accomplish those objectives.

Policy and FinOps

FinOps,[65] or the application of DevOps principles to cost control efforts, is a key use case for the policy engine and one that it is specifically designed to address. Large enterprises require hefty budgets to continue to produce income in their vertical, but leakage of resources reduces a company's velocity and in extreme cases can lead to failure. Technology assets are notoriously difficult to catalog and maintain in such a way that their expense can be mapped directly to the profits they generate, and this challenge only becomes more complex when running in a multi-cloud or hybrid environment. Providing a true accounting of what is in use and assuring usage is trimmed to avoid waste is an area where policy controls shine. Per Tarun Sood,[66] FinOps encourages a culture of accountability

[65] https://learn.microsoft.com/en-us/azure/cost-management-billing/finops/overview-finops

[66] www.linkedin.com/pulse/demystifying-azure-cloud-finops-best-practices-optimizing-tarun-sood/

through "clear cost allocation policies that outline the responsibilities and expectations of different teams or departments in managing their Azure costs."

Policies can govern the type of resource or service that can be created and by whom; they can enforce tagging to ensure that costs are easily segregated by department or workload, they can specify which product licenses are available, they can enforce scaling limits, and much more. Key cost management risks include not only overspending but underutilization of capacity blocks that may have been purchased for a discounted cloud spend, and policy can help you control these risks by either remediating policy exceptions or providing timely and actionable audit information. Utilization also matters as companies pursue ESG objectives, and providing sufficient capacity for client-facing workloads to perform well demonstrates that FinOps policy can be used not only to control costs but to increase profits. Every Arc-enabled Kubernetes cluster in Azure has a unique Azure Resource Manager ID and is contained within a subscription and Resource Group, enabling you to both monitor and proscribe how those resources are used with policy.

Effective Policy

Policy is the most effective tool available to assure your governance intentions become reality in your systems. Its design and implementation are as important as the design of the systems and workloads themselves in order to assure the latter two remain operational. We could coin YAOT [Yet Another Ops Term] such as PolSecOps, and it might stick, but the better approach is to simply view policy as the natural approach to governance and then practice governance faithfully starting far left.

A 2022 Trend Micro blog post[67] addresses a perhaps overlooked aspect of policy management and security – finding Open Policy Agent servers open to the Internet – and comments, "If OPA servers are left unsecured, their policies can reveal sensitive application information, including user profiles and services being used. These exposed policies can also unwittingly disclose information on how the system should behave to bypass restrictions on implemented policies, which malicious actors can use to wage attacks." As you would secure any other part of your supply chain, lock down your policies.

If you are a policy geek, you will enjoy digging into the personal project site[68] of a Microsoft employee that attempts to index and update all of the built-in policies that are available. Next, we'll look at process automation underpinned by a monitoring feedback loop and what's possible in terms of automated response.

[67] www.trendmicro.com/en_fi/research/22/h/what-exposed-opa-servers-can-tell-you-about-your-applications-.html

[68] www.azadvertizer.net/index.html