DevOps

Week 15

Murtaza Munawar Fazal

Implement pipeline security

- It's fundamental to protect your code protecting credentials, and secrets. The following list is several operational practices that a team ought to apply to protect itself:
    - Authentication and authorization.
        - Use multifactor authentication (MFA) and
        - Just-In-Time administration tools such as Azure PowerShell Just Enough Administration (JEA)
    - The CI/CD Release Pipeline.
        - If the release pipeline and cadence are damaged, use this pipeline to rebuild infrastructure. Manage Infrastructure as Code (IaC) with Azure Resource Manager or use the Azure platform as a service (PaaS) or a similar service.
    - Permissions management.
        - You can manage permissions to secure the pipeline with role-based access control (RBAC), just as you would for your source code.

Implement pipeline security

- Dynamic scanning.
  - It's the process of testing the running application with known attack patterns. You could implement penetration testing as part of your release.
- Production monitoring.
  - It's a critical DevOps practice. The specialized services for detecting anomalies related to intrusion are known as Security Information and Event Management. Microsoft Defender for Cloud focuses on the security incidents related to the Azure cloud.

Microsoft Defender for Cloud

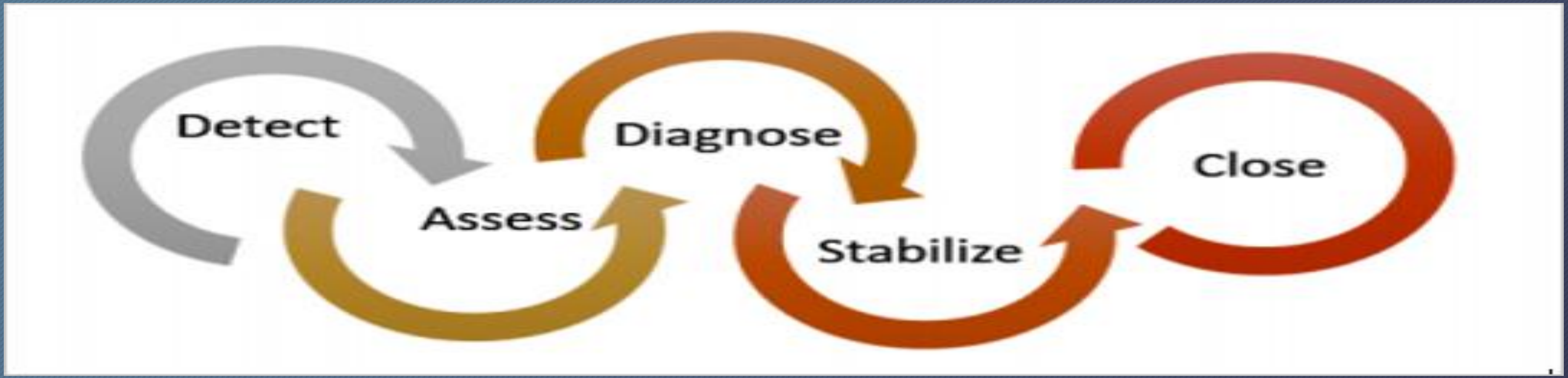
- Microsoft Defender for Cloud is a monitoring service that provides threat protection across all your services both in Azure and on-premises. Microsoft Defender can:
  - Provide security recommendations based on your configurations, resources, and networks.
  - Monitor security settings across on-premises and cloud workloads and automatically apply required security to new services as they come online.
  - Continuously monitor all your services and do automatic security assessments to identify potential vulnerabilities before they can be exploited.
  - Use Azure Machine Learning to detect and block malicious software from being installed on your virtual machines (VMs) and services. You can also define a list of allowed applications to ensure that only the validated apps can execute.
  - Analyze and identify potential inbound attacks and help investigate threats and any post-breach activity that might have occurred.
  - Provide just-in-time (JIT) access control for ports by reducing your attack surface by ensuring the network only allows traffic that you require.

Microsoft Defender for Cloud versions

- Microsoft Defender for Cloud supports both Windows and Linux operating systems. It can also provide security to features in IaaS and platform as a service (PaaS) scenarios.
- Microsoft Defender for Cloud is available in two versions:
  - Free. Available as part of your Azure subscription, this tier is limited to assessments and Azure resources' recommendations only.
  - Standard. This tier provides a full suite of security-related services, including continuous monitoring, threat detection, JIT access control for ports, and more.

Microsoft Defender for Cloud usage scenarios

- You can integrate Microsoft Defender for Cloud into your workflows and use it in many ways. For example, use Microsoft Defender for Cloud as part of your incident response plan.
- Many organizations only respond to security incidents after an attack has occurred. To reduce costs and damage, it's necessary to have an incident response plan before an attack occurs.



Microsoft Defender for Cloud usage scenarios

- The following examples show how to use Microsoft Defender for Cloud to detect, assess, and diagnose your incident response plan stages.
  - Detect. Review the first indication of an event investigation.
    - For example, use the Microsoft Defender for Cloud dashboard to review a high-priority security alert's initial verification.
  - Assess. Do the initial assessment to obtain more information about suspicious activity.
    - For example, you can get more information about a security alert from Microsoft Defender for Cloud.
  - Diagnose. Conduct a technical investigation and identify containment, mitigation, and workaround strategies.
    - For example, you can follow the remediation steps described by Microsoft Defender for Cloud for a particular security alert.
  - Use Microsoft Defender for Cloud recommendations to enhance security.

Microsoft Defender for Cloud usage scenarios

- You can reduce the chances of a significant security event by configuring a security policy and then implementing the recommendations provided by Microsoft Defender for Cloud.
- A security policy defines the controls recommended for resources within a specified subscription or resource group. You can define policies in Microsoft Defender for Cloud according to your company's security requirements.
- Microsoft Defender for Cloud analyzes the security state of your Azure resources. When identifying potential security vulnerabilities, it creates recommendations based on the controls set in the security policy.
- For example, if you have workloads that don't require the Azure SQL Database Transparent Data Encryption (TDE) policy, turn off the policy at the subscription level and enable it only on the resource groups where SQL Database TDE is required.

Azure Policy

- Azure Policy is an Azure service that you can create, assign, and manage policies.
- Policies enforce different rules and effects over your Azure resources, ensuring that your resources stay compliant with your standards and SLAs.
- Azure Policy uses policies and initiatives to provide policy enforcement capabilities.
- For example, you might have a policy that specifies a maximum size limit for VMs in your environment.

Azure Policy

- After you implement your maximum VM size policy, Azure Policy will evaluate the VM resource whenever a VM is created—or updated to ensure that the VM follows the size limit you set in your Policy.
- Azure Policy can help maintain the state of your resources by evaluating your existing resources and configurations and automatically remediating non-compliant resources.
- Azure Policy can also integrate with Azure DevOps by applying any continuous integration (CI) and continuous delivery (CD) pipeline policies that apply to the pre-deployment and post-deployment of your applications. Using Azure policies, Check gate provides security and compliance assessment on the resources with an Azure resource group or subscription that you can specify.

Azure Policy

- Applying a policy to your resources with Azure Policy involves the following high-level steps:
  - Policy definition. Create a policy definition.
  - Policy assignment. Assign the definition to a scope of resources.
  - Remediation. Review the policy evaluation results and address any non-compliances.

Policy definition

- A policy definition specifies the resources to be evaluated and the actions to take on them. Policies are defined in the JavaScript Object Notation (JSON) format.
- The following example defines a policy that limits where you can deploy resources:

```
{
    "properties": {
        "mode": "all",
        "parameters": {
            "allowedLocations": {
                "type": "array",
                "metadata": {
                    "description": "The list of locations that can be specified when deploying resources",
                    "strongType": "location",
                    "displayName": "Allowed locations“ }  }
        },
        "displayName": "Allowed locations",
        "description": "This policy enables you to restrict the locations your organization can specify when
deploying resources.",
        "policyRule": {
            "if": {
                "not": {
                    "field": "location",
                    "in": "[parameters('allowedLocations')]"
                }
            },
            "then": {
                "effect": "deny"
            }  }  }  }
```

Policy assignment

- Policy definitions, whether custom or built-in, need to be assigned.
- A policy assignment is a policy definition that has been assigned to a specific scope. Scopes can range from a management group to a resource group.
- Child resources will inherit any policy assignments applied to their parents.
- It means if a policy is applied to a resource group, it's used to all the resources within that resource group.
- However, you can define subscopes for excluding resources from policy assignments.
- You can assign policies via:
    - Azure portal.
    - Azure CLI.
    - PowerShell.

Remediation

- Resources found not to follow a deployIfNotExists or modify policy condition can be put into a compliant state through Remediation.
- Remediation instructs Azure Policy to run the deployIfNotExists effect or the tag operations of the policy on existing resources.
- To minimize configuration drift, you can bring resources into compliance using automated bulk Remediation instead of going through them one at a time.
- You can read more about Azure Policy on the Azure Policy webpage.

Resource Locks

- Locks help you prevent accidental deletion or modification of your Azure resources.
- In the Azure portal, locks are called Delete and Read-only, respectively.
- You might need to lock a subscription, resource group, or resource to prevent users from accidentally deleting or modifying critical resources.
- You can set a lock level to CanNotDelete or ReadOnly:
  - CanNotDelete means that authorized users can read and modify a resource, but they can't delete it.
  - ReadOnly means that authorized users can read a resource, but they can't modify or delete it.

Explore Azure Blueprints

- Azure Blueprints enables cloud architects to define a repeatable set of Azure resources that implement and adhere to an organization's standards, patterns, and requirements.
- Azure Blueprints helps development teams build and deploy new environments rapidly with a set of built-in components that speed up development and delivery.
- Furthermore, it's done while staying within organizational compliance requirements.
- Azure Blueprints provides a declarative way to orchestrate deployment for various resource templates and artifacts, including:
  - Role assignments
  - Policy assignments
  - Azure Resource Manager templates
  - Resource groups

Explore Azure Blueprints

- To implement Azure Blueprints, complete the following high-level steps:
  1. Create a blueprint.
  2. Assign the blueprint.
  3. Track the blueprint assignments.

- With Azure Blueprints, the relationship between the blueprint definition (what should be deployed) and the blueprint assignment (what is deployed) is preserved.
- The blueprints in Azure Blueprints are different from Azure Resource Manager templates.
- When Azure Resource Manager templates deploy resources, they have no active relationship with the deployed resources.
- By contrast, with Azure Blueprints, each deployment is tied to an Azure Blueprints package. It means that the relationship with resources will be maintained, even after deployment. This way, keeping relationships improves deployment tracking and auditing capabilities.

Configuration information in files

- Most application runtime environments include configuration information that is held in files deployed with the application.
- In some cases, it's possible to edit these files to change the application behavior after deploying.
- However, changes to the configuration require the application to be redeployed, often resulting in unacceptable downtime and other administrative overhead.
- Local configuration files also limit the configuration to a single application, but sometimes it would be helpful to share configuration settings across multiple applications.
- Examples include database connection strings.

Configuration information in files

- It's challenging to manage changes to local configurations across multiple running instances of the application, especially in a cloud-hosted scenario.
- It can result in instances using different configuration settings while the update is being deployed. Also, updates to applications and components might require changes to configuration schemas.

History of Configuration Files

- Since the early days of .NET, the app.config and web.config files have the notion that developers can make their code flexible by moving typical configuration into these files.
- When used effectively, these files are proven to be worthy of dynamic configuration changes. However, much time, we see the misuse of what goes into these files.
- A common culprit is how samples and documentation have been written. Most samples on the web would usually use these config files to store key elements such as ConnectionStrings and even passwords.
- The values might be obfuscated but what we are telling developers is that "hey, It's a great place to push your secrets!".
- So, in a world where we're preaching using configuration files, we can't blame the developer for not managing its governance.
- Configs are great for ensuring the flexibility of the application, config files. However, they aren't straightforward, especially across environments.

A ray of hope: The DevOps movement

- In recent years, we've seen a shift around following some excellent practices around effective DevOps and some great tools (Chef, Puppet) for managing Configuration for different languages.
- While these have helped inject values during CI/CD pipeline and greatly simplified configuration management, the blah.config concept hasn't moved away.
- Frameworks like ASP.NET Core support the notion of appSettings.json across environments.

Separation of concerns

- One of the key reasons we would want to move the configuration away from source control is to outline responsibilities.
  - Configuration custodian:
    - Responsible for generating and maintaining the life cycle of configuration values. These include CRUD on keys, ensuring the security of secrets, regeneration of keys and tokens, defining configuration settings such as Log levels for each environment.
    - This role can be owned by operation engineers and security engineering while injecting configuration files through proper DevOps processes and CI/CD implementation. They do not define the actual configuration but are custodians of their management.

Separation of concerns

- Configuration consumer:
  - Responsible for defining the schema (loose term) for the configuration that needs to be in place and then consuming the configuration values in the application or library code.
  - It's the Dev. And Test teams shouldn't be concerned about the value of keys but rather what the key's capability is. For example, a developer may need a different ConnectionString in the application but not know the actual value across different environments.
- Configuration store:
  - The underlying store used to store the configuration, while it can be a simple file, but in a distributed application, it needs to be a reliable store that can work across environments. The store is responsible for persisting values that modify the application's behavior per environment but aren't sensitive and don't require any encryption or HSM modules.

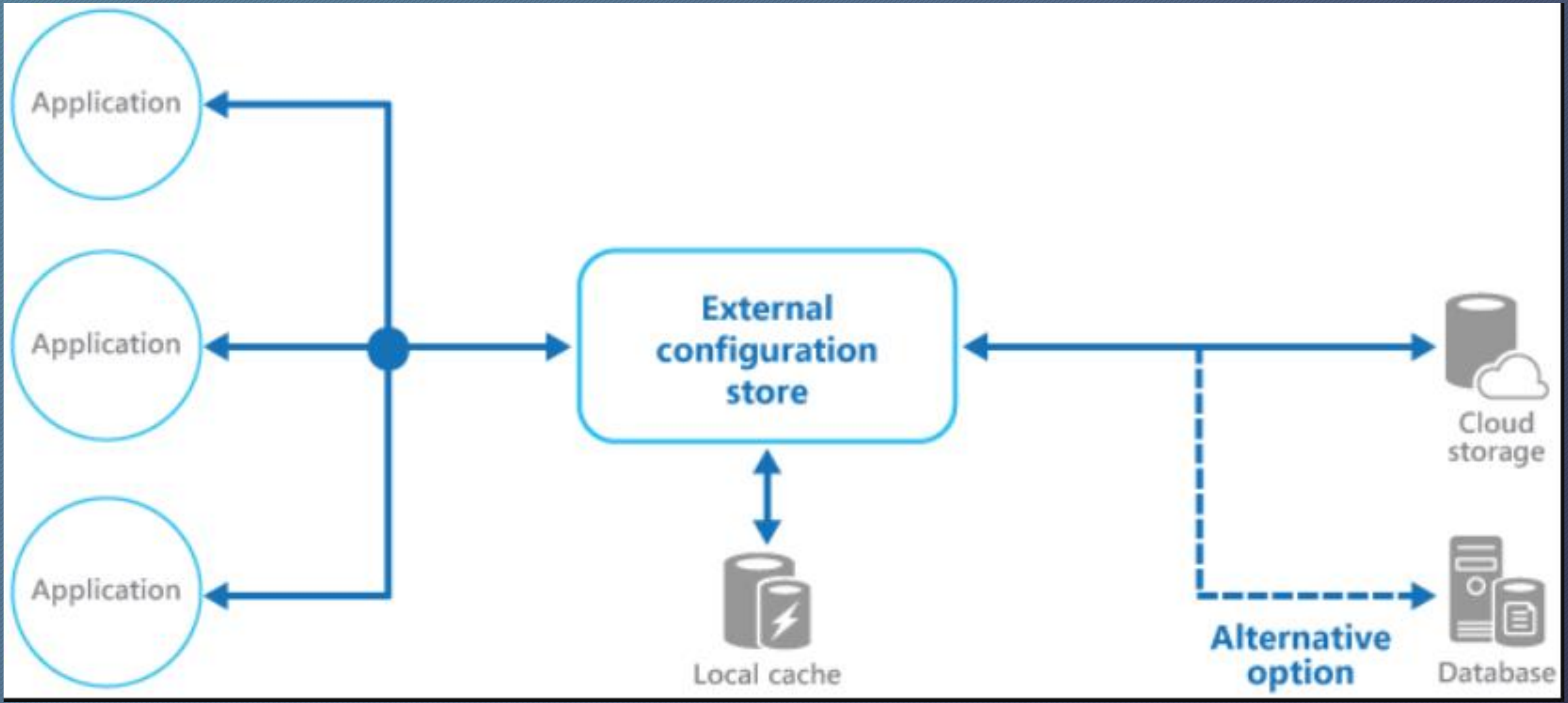
Separation of concerns

- Secret store:
  - While you can store configuration and secrets together, it violates our separation of concern principle, so the recommendation is to use a different store for persisting secrets.
  - It allows a secure channel for sensitive configuration data such as ConnectionStrings, enables the operations team to have Credentials, Certificate, Token in one repository, and minimizes the security risk if the Configuration Store gets compromised.

Understand external configuration store patterns

- These patterns store the configuration information in an external location and provide an interface that can be used to quickly and efficiently read and update configuration settings.
- The type of external store depends on the hosting and runtime environment of the application.
- A cloud-hosted scenario is typically a cloud-based storage service but could be a hosted database or other systems.
- The backing store you choose for configuration information should have an interface that provides consistent and easy-to-use access.
- It should expose the information in a correctly typed and structured format.
- The implementation might also need to authorize users' access to protect configuration data and be flexible enough to allow storage of multiple configuration versions (such as development, staging, or production, including many release versions of each one).
- Many built-in configuration systems read the data when the application starts up and cache the data in memory to provide fast access and minimize the impact on application performance.

Understand external configuration store patterns



Azure App Configuration

- Azure App Configuration is a service for central management of application settings and feature flags.
- Distributed configuration settings can lead to hard-to-troubleshoot deployment errors.
- Azure App Configuration service stores all the settings for your application and secures their access in one place.
- Azure App Configuration service provides the following features:
    - A fully managed service that can be set up in minutes.
    - Flexible key representations and mappings.
    - Tagging with labels.
    - A point-in-time replay of settings.
    - Dedicated UI for feature flag management.
    - Comparison of two sets of configurations on custom-defined dimensions.
    - Enhanced security through Azure managed identities.
    - Complete data encryptions, at rest or in transit.
    - Native integration with popular frameworks.

Azure App Configuration

- App Configuration complements Azure Key Vault, which is used to store application secrets. App Configuration makes it easier to implement the following scenarios:
  - Centralize management and distribution of hierarchical configuration data for different environments and geographies.
  - Dynamically change application settings without the need to redeploy or restart an application.
  - Control feature availability in real time.

Azure Key Vault with Azure Pipelines

- Applications contain many secrets, such as:
  - Connection strings.
  - Passwords.
  - Certificates.
  - Tokens, which, if leaked to unauthorized users, can lead to a severe security breach.
- It can result in severe damage to the organization's reputation and compliance issues with different governing bodies.
- Azure Key Vault allows you to manage your organization's secrets and certificates in a centralized repository.
- The secrets and keys are further protected by Hardware Security Modules (HSMs).
- It also provides versioning of secrets, full traceability, and efficient permission management with access policies.