

# DevOps Introduction

By: Rohit K Singh



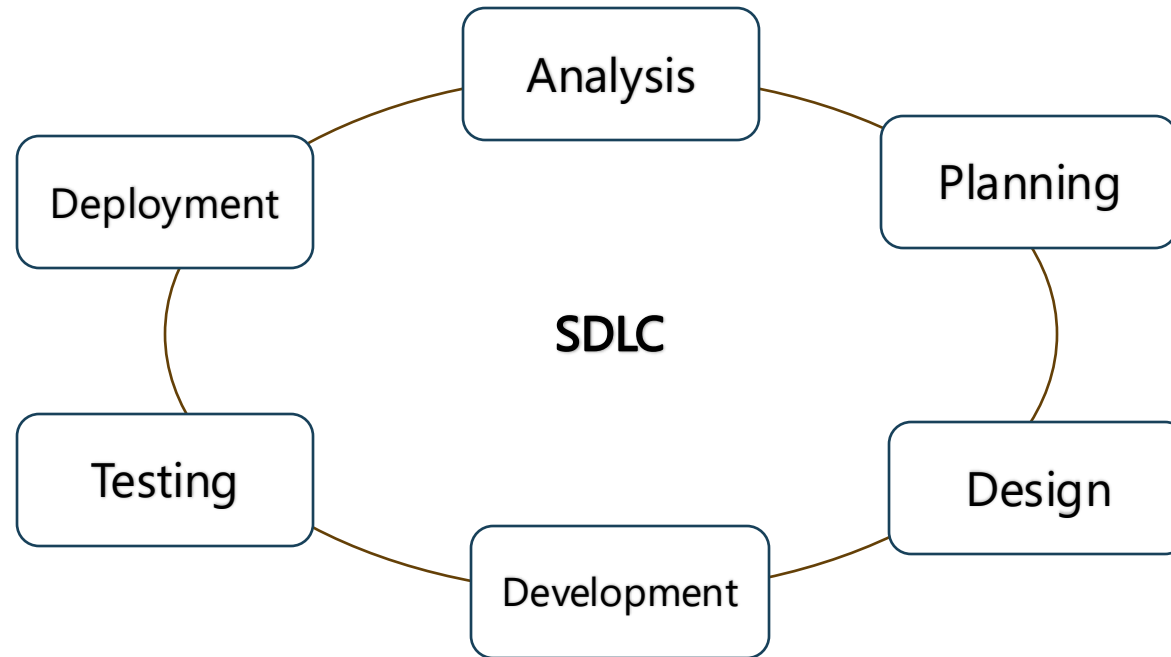
# Agenda

- ❑ Software Development Life Cycle
- ❑ Software Development Models
  - Waterfall Model
  - Agile Methodology
  - SCRUM Methodology
  - Kanban Overview
- ❑ DevOps Overview
- ❑ DevOps Lifecycle



# SDLC:

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software



- ✓ Lowering COST
- ✓ Highest Quality
- ✓ Shortened Overall Time



# Before DevOps:

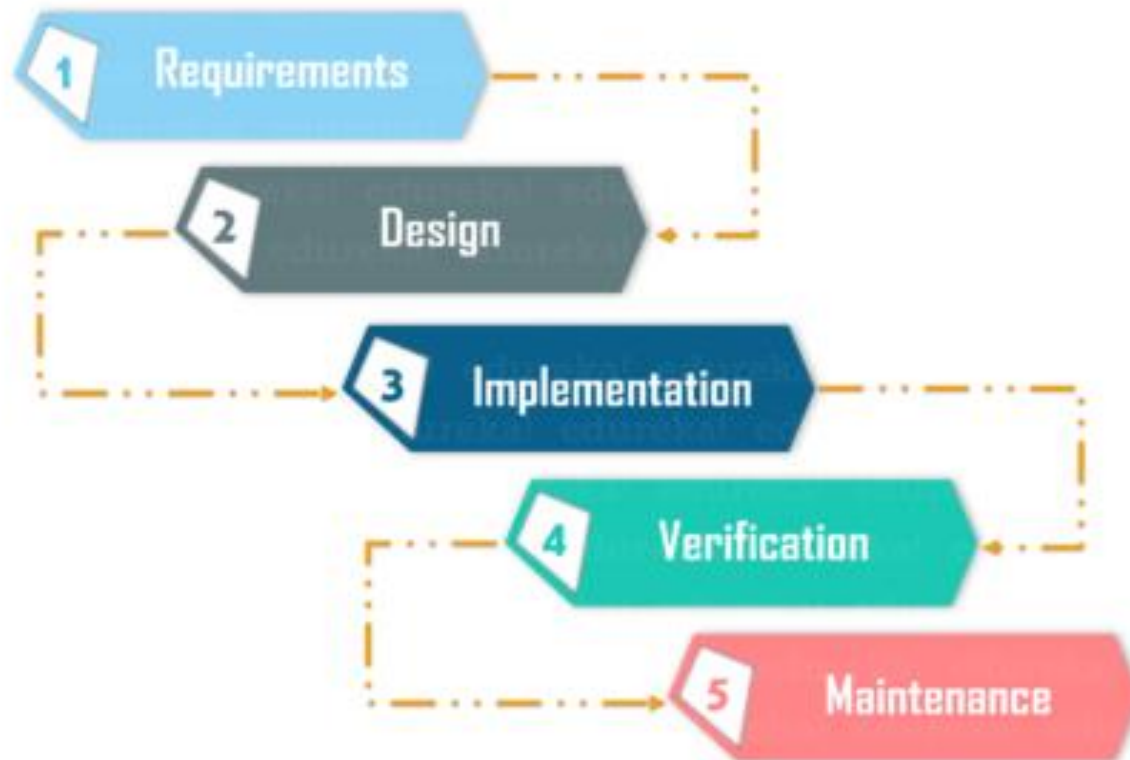
- ❑ Waterfall Model

- ❑ Agile Methodology



# Waterfall Model:

- The waterfall model is a software development model that is straightforward and linear
- It follows a top-down approach



# Agile Methodology:

- Agile Methodology is an iterative based software development approach It follows a top-down approach
- The software project is broken down into various iterations or sprints
- Each iteration has phases like the waterfall model such as Requirements Gathering, Design, Development, Testing, and Maintenance
- The duration of each iteration is generally 2-8 weeks



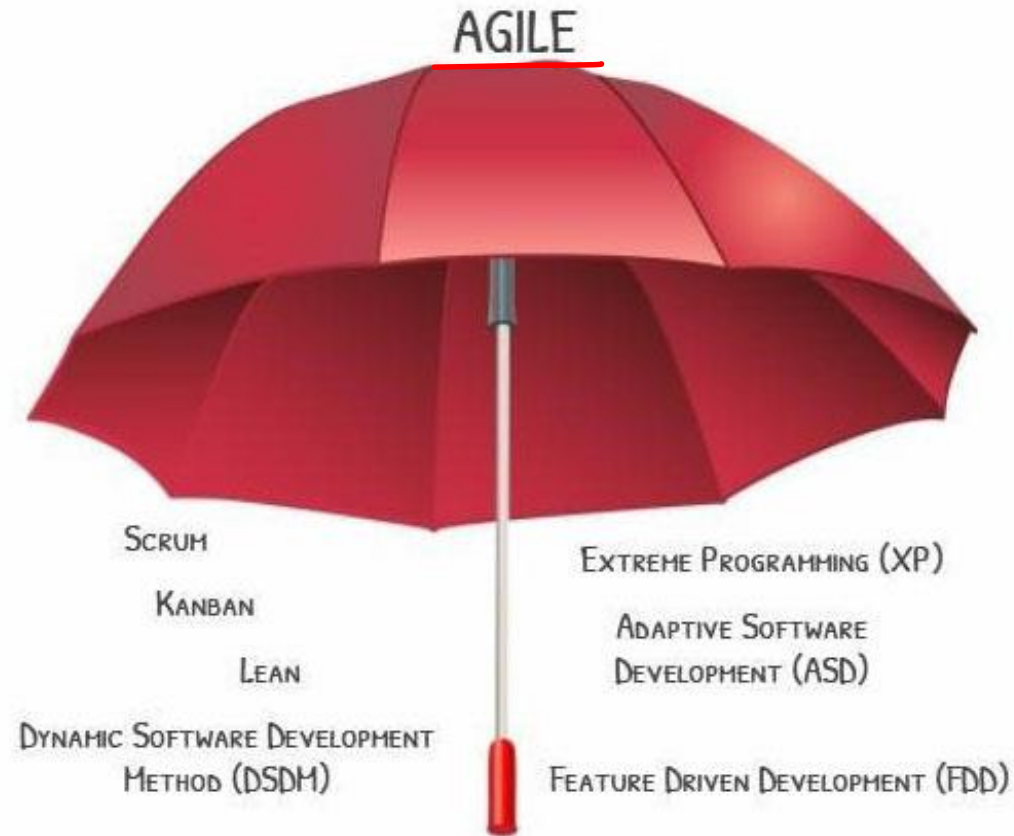
- ✓ High priority features
- ✓ Customer feedback on application performance
- ✓ Application improvisation post-feedback
- ✓ Repeat the entire process for the desired software quality



# Key Agile Methodology:

Agile is an umbrella term for several methods and practices. Let's look at some of the popular methodologies:

- Scrum
- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Software Development Method (DSDM)
- Feature Driven Development (FDD)
- Kanban
- Behaviour Driven Development (BDD)



# 12 Agile Principles:



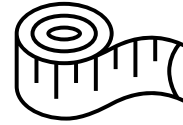
Customer  
satisfactions



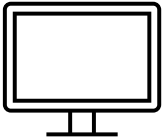
Communicate  
regularly



Measure  
progress



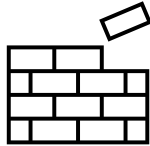
Measure  
work progress



Changing  
requirements



Support  
team members



Development  
process



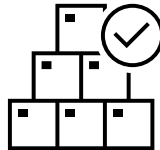
Continue  
Seeking result



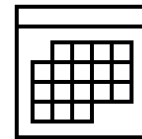
Frequent  
delivery



Face - to - Face  
communication



Good  
design



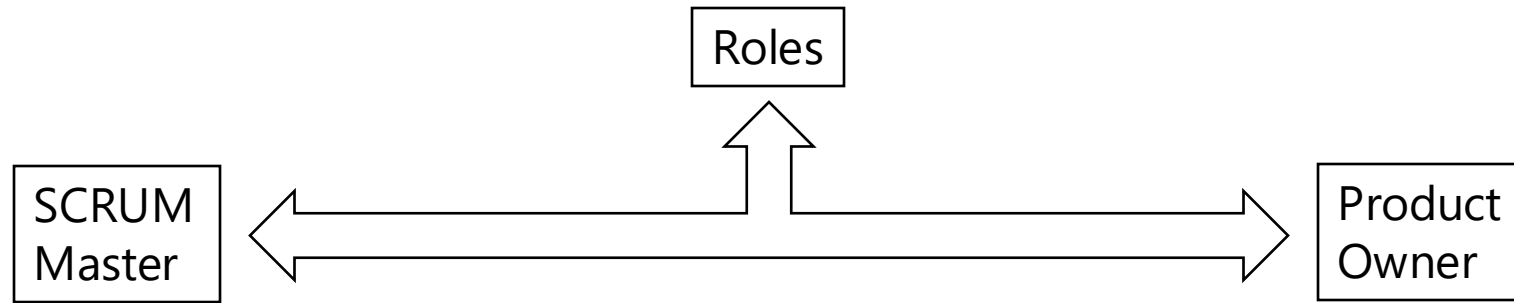
Reflect and  
adjust regularly





# SCRUM Methodology:

- Scrum is one of the more popular and widely used Agile frameworks

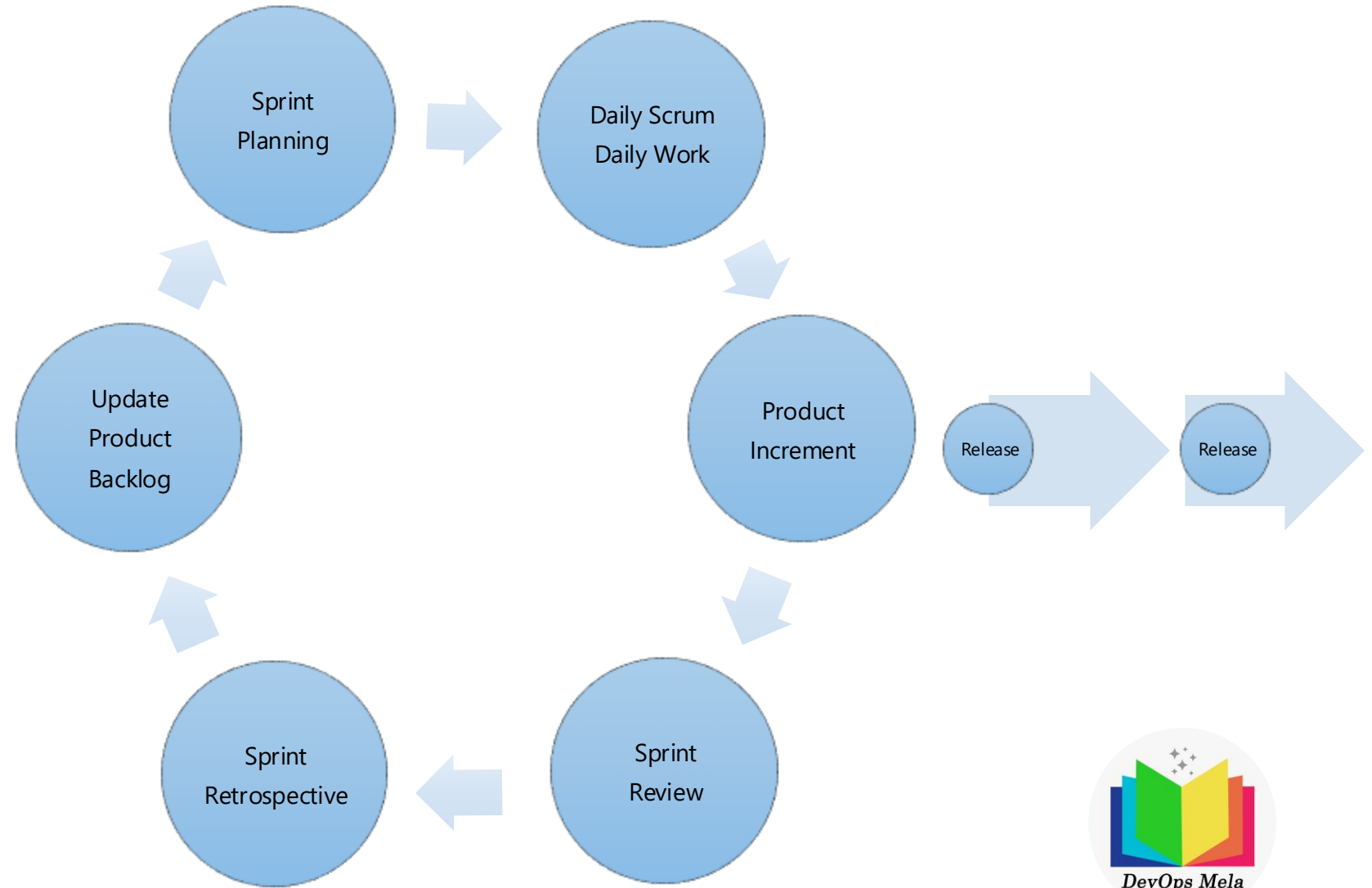
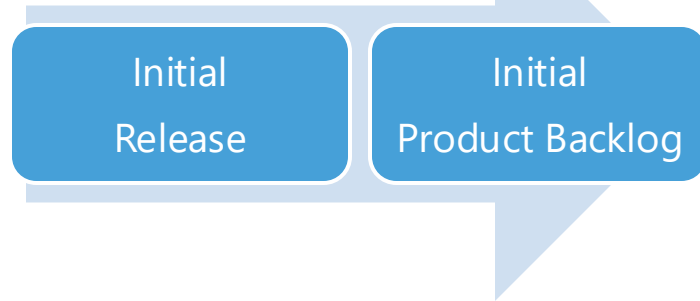
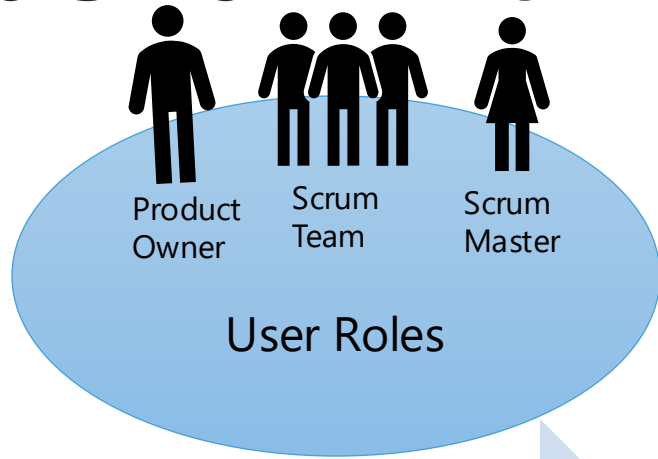


- Scrum Master is the coach and the gatekeeper
- Scrum Master establishes responsibility for following the Agile framework providing guidance and education to the Scrum Team
- Removing impediments and distractions

- The Product Owner of your Scrum Team is first and foremost the subject matter expert
- The Product Owner keeps track of the projects' stakeholders' expectations and defines and gathers the required tools and resources that the Scrum Team needs
- The Product Owner communicates their vision to the team to help set priorities

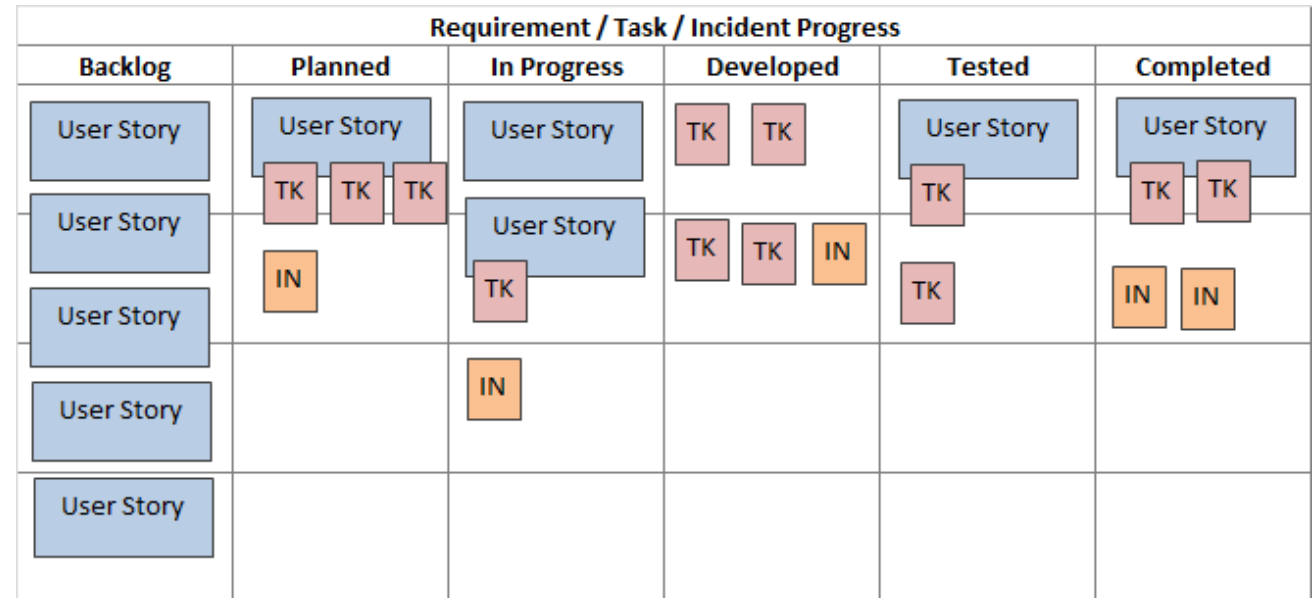


# SCRUM Workflow:



# Kanban Methodology:

- Kanban is a scheduling system for lean and other JIT processes
- There are physical (or virtual) "cards" called Kanban that move through the process from start to finish
- The aim is to keep a constant flow of Kanban so that as inventory is required at the end of the process, just that much is created at the start



# DevOps Overview

- DevOps is a culture where collaboration between development, operations, and business teams is considered a critical aspect of the DevOps journey
- It's not solely about the tools and DevOps in an organization that creates continuous value for customers
- Tools are only one of the pillars, the other being People and Processes
- DevOps focuses on releasing small features regularly
  - ✓ Better software quality
  - ✓ Quick client feedback
  - ✓ High customer satisfaction

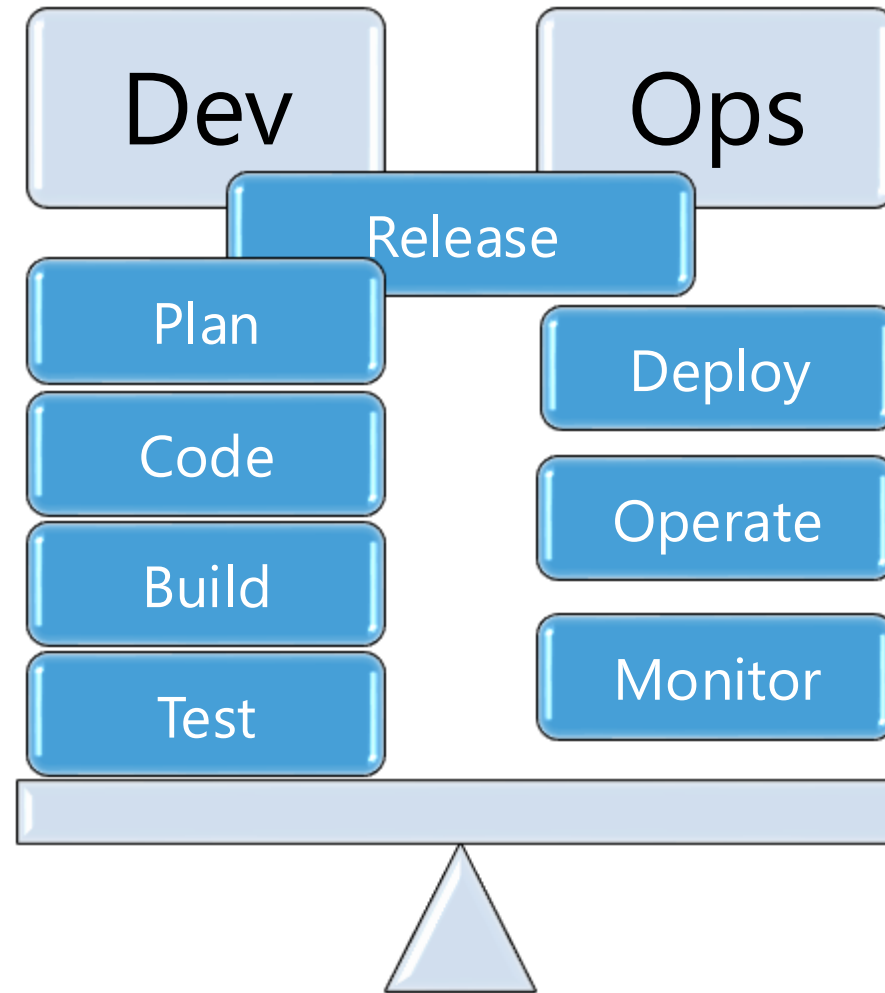


# DevOps Lifecycle:

- Continuous Planning
- Collaborative Deployment
- Continuous Testing
- Continuous Release and Deployment
- Continuous Monitoring
- Continuous Feedback and Optimization

## **Benefits of DevOps:**

- Predictability
- Maintainability
- Reproducibility
- Greater Quality
- Time to market
- Reduced Risk
- Cost Efficiency
- Resiliency



# Summary:

## SDLC

- Analysis
- Planning
- Design
- Development
- Testing
- Deployment

## Methodology:

- Waterfall
- Agile
- Scrum
- Kanban

- DevOps Overview
- DevOps Lifecycle



# Azure DevOps Introduction

By: Rohit K Singh



# Agenda:

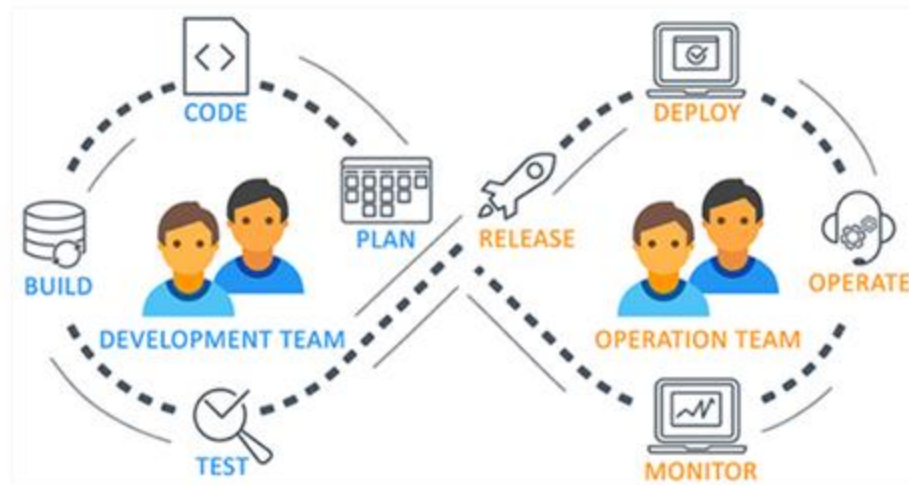
- ❑ Introduction to Azure DevOps
- ❑ Azure Boards - Overview
- ❑ Azure Boards Process
  - Basic
  - Agile
  - SCRUM
  - CMMI
- ❑ Demo





# Azure DevOps:

- DevOps is the union of people, processes, and products to enable continuous delivery of value to the end users
- Using Azure DevOps, we can plan smarter, collaborate better, and deliver reliable software faster with a set of modern services



☐ Azure Boards

☐ Azure Pipelines

☐ Azure Repos

☐ Azure Test Plans

☐ Azure Artifacts

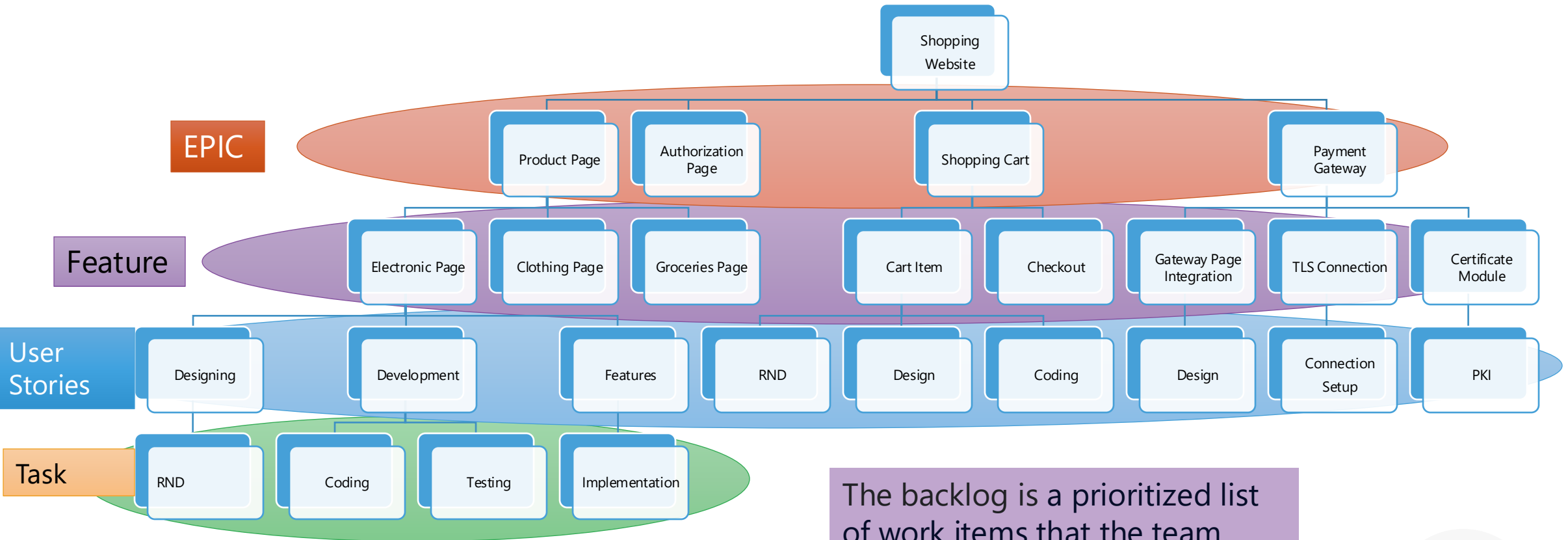


# Azure Boards:

- Azure Boards is a web-based service that enables teams to plan, track, and discuss work across the entire development process, while it supports agile methodologies, including Scrum and Kanban
- Azure Boards provides a customizable platform for managing work items, allowing teams to collaborate effectively and streamline their workflow

Azure Boards (hub)	Functions
Work Items	Access lists of work items based on specific criteria, such as those assigned to you, ones you follow, and work items you viewed or updated
Boards	View work items as cards and update their status through drag-and-drop, Use this feature to implement Kanban practices and visualize workflow for a team
Backlogs	View, plan, order, and organize work items
Sprints	Access your team's filtered view of work items based on a specific sprint or iteration path
Queries	Generate custom work item lists and perform various tasks, such as triage work, make bulk updates, and view relationships between work items
Delivery Plans	Management teams can view deliverables and track dependencies across multiple teams in a calendar view
Analytics Views	Create highly sophisticated Power BI reports, based on Azure Boards data (work items). Access default Analytics views or create a custom view

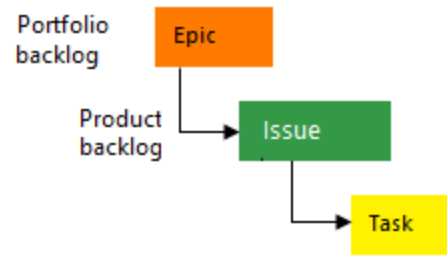
# Use-case {Backlog}:



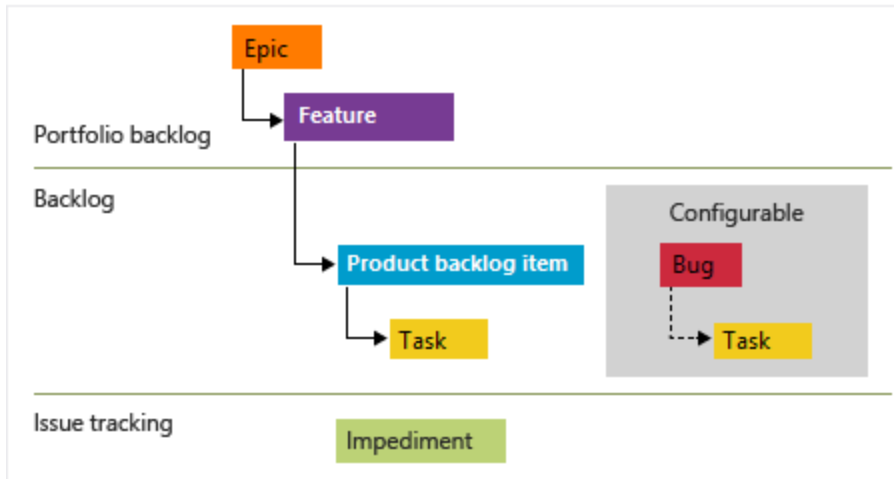
The backlog is a prioritized list of work items that the team plans to complete in the future



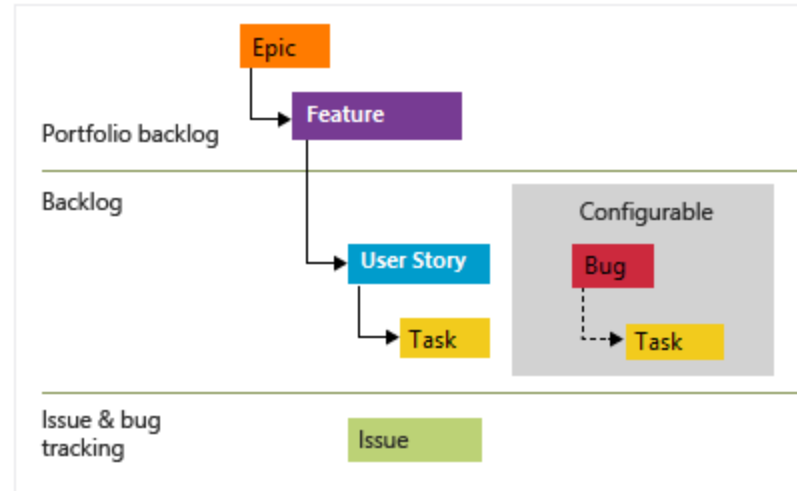
# Work Item Types:



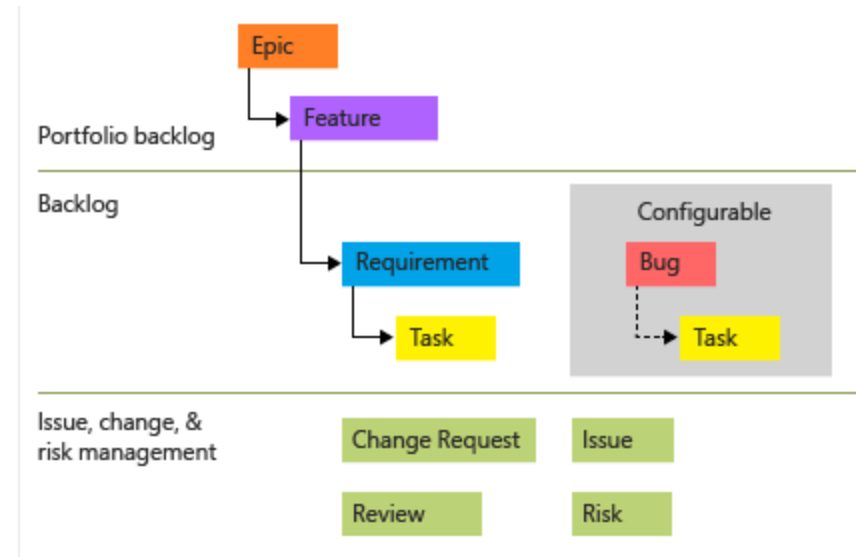
## Basic:



## SCRUM:



## Agile:



## CMMI: (Capability Maturity Model Integration)



# Work Item (Charts):

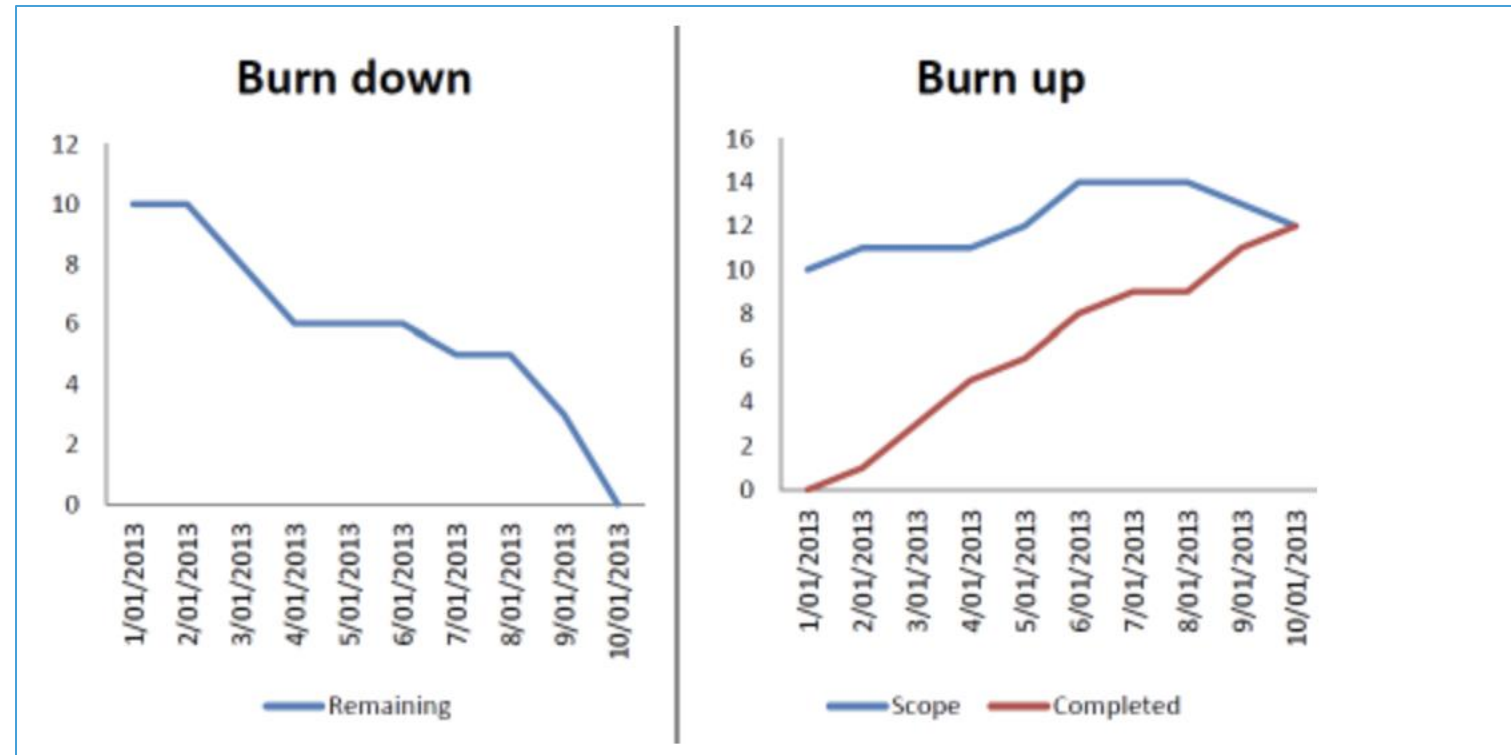
- Burn-down and burn-up charts are two types of charts that project managers use to track and communicate the progress of their projects

## ☐ Burndown chart:

A burn-down chart shows how much work is remaining to be done in the project

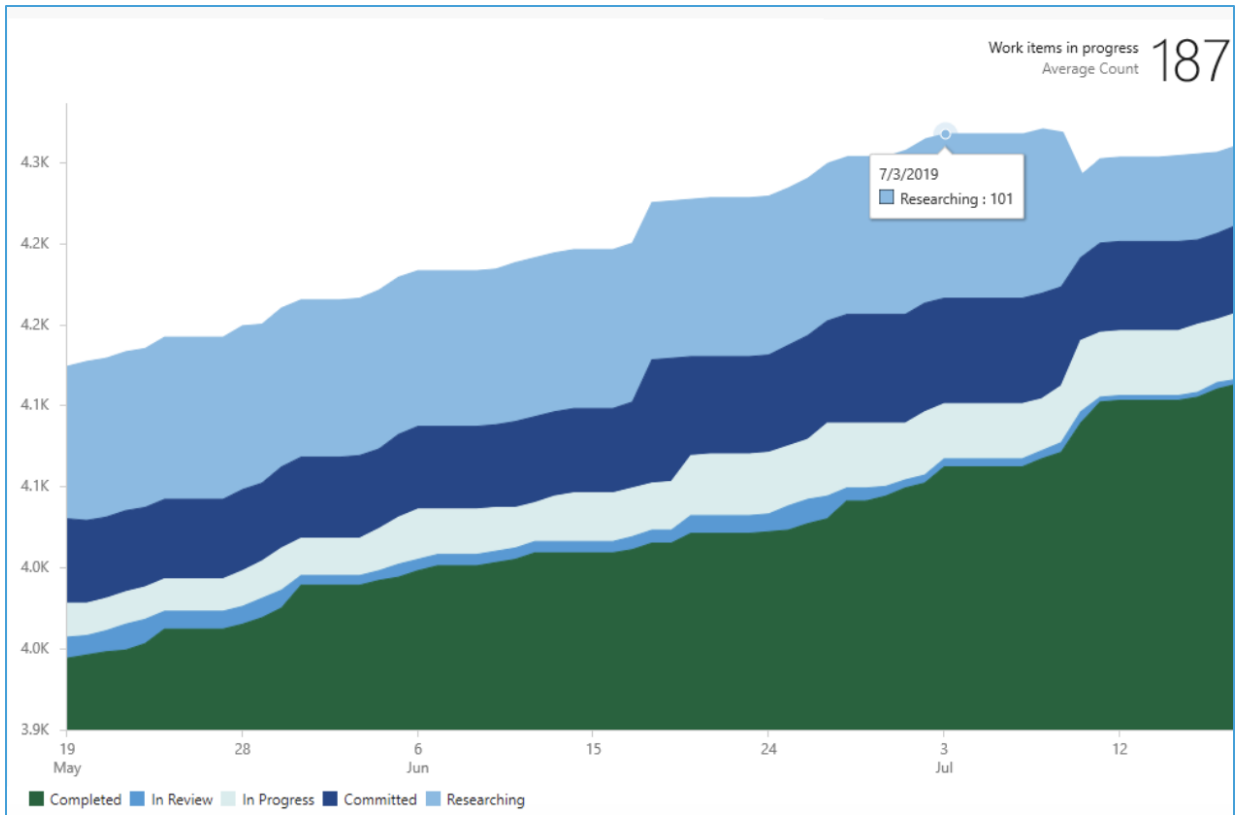
## ☐ Burnup chart:

A burn-up shows how much work has been completed, and the total amount of work



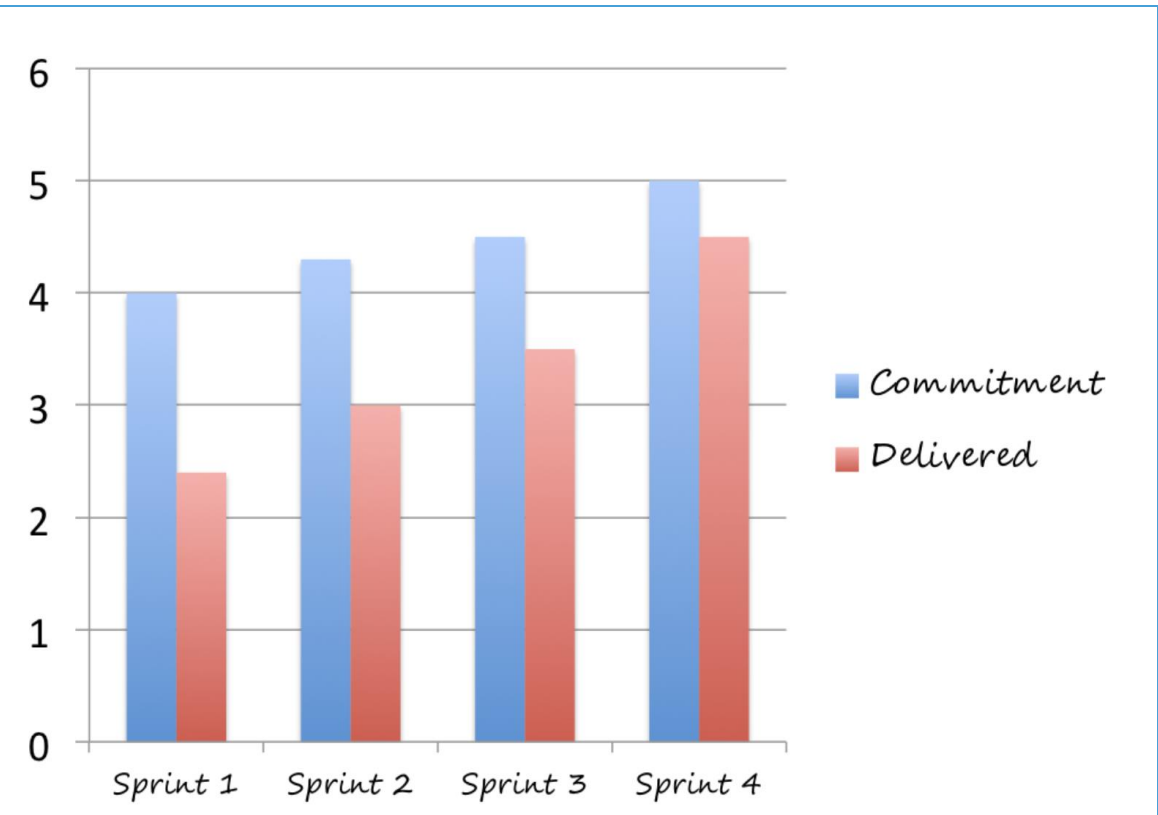
❑ Cumulative flow diagram:

Use cumulative flow diagrams (CFD) to monitor the flow of work and identify bottlenecks in the software development process



❑ Velocity:

The main purpose of the velocity chart is to overview how much work has been delivered for each sprint. It will help you to have a clear view of future perspectives and the workload of your team



# DEMO



# Summary:

- Introduction to Azure DevOps
- Various Azure DevOps components

## Azure Boards (Process):

- Basic
- SCRUM
- Agile
- CMMI





# Repository Management

By: Rohit K Singh



# Agenda:

- ❑ Introduction to Version Control
- ❑ Different types of Version Control
  - Centralized
  - Distributed
- ❑ Introduction to GIT
- ❑ Understanding GIT Stages
- ❑ Installation and Setup
- ❑ DEMO

- ❑ Introduction to Azure Repos
- ❑ Branching and merging strategy
- ❑ Versioning
- ❑ Different merge-types in Azure Repos
  - Merge (no fast forward)
  - Squash
  - Rebase and fast-forward.
  - Semi-linear merge

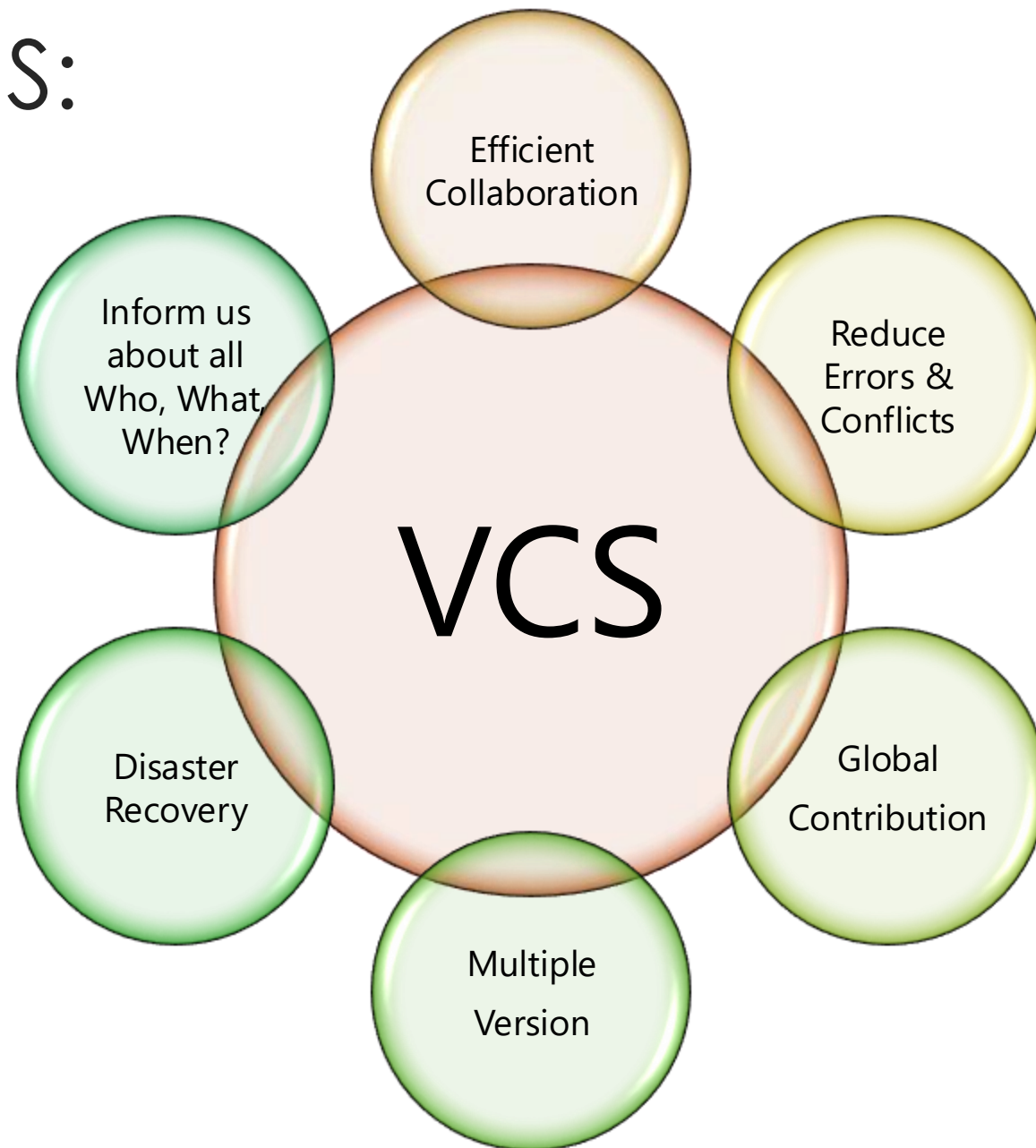


# Version Control?

- ☐ Version control, also known as source control, is the practice of tracking and managing changes to software code
- ☐ Version control software keeps track of every modification to the code in a special kind of database
- ☐ Version control tracks every developer change and helps prevent concurrent work from conflicting



# Benefits of VCS:



# Types of Version Control:

- ❑ Centralized Version Control

- ❑ Distributed Version Control

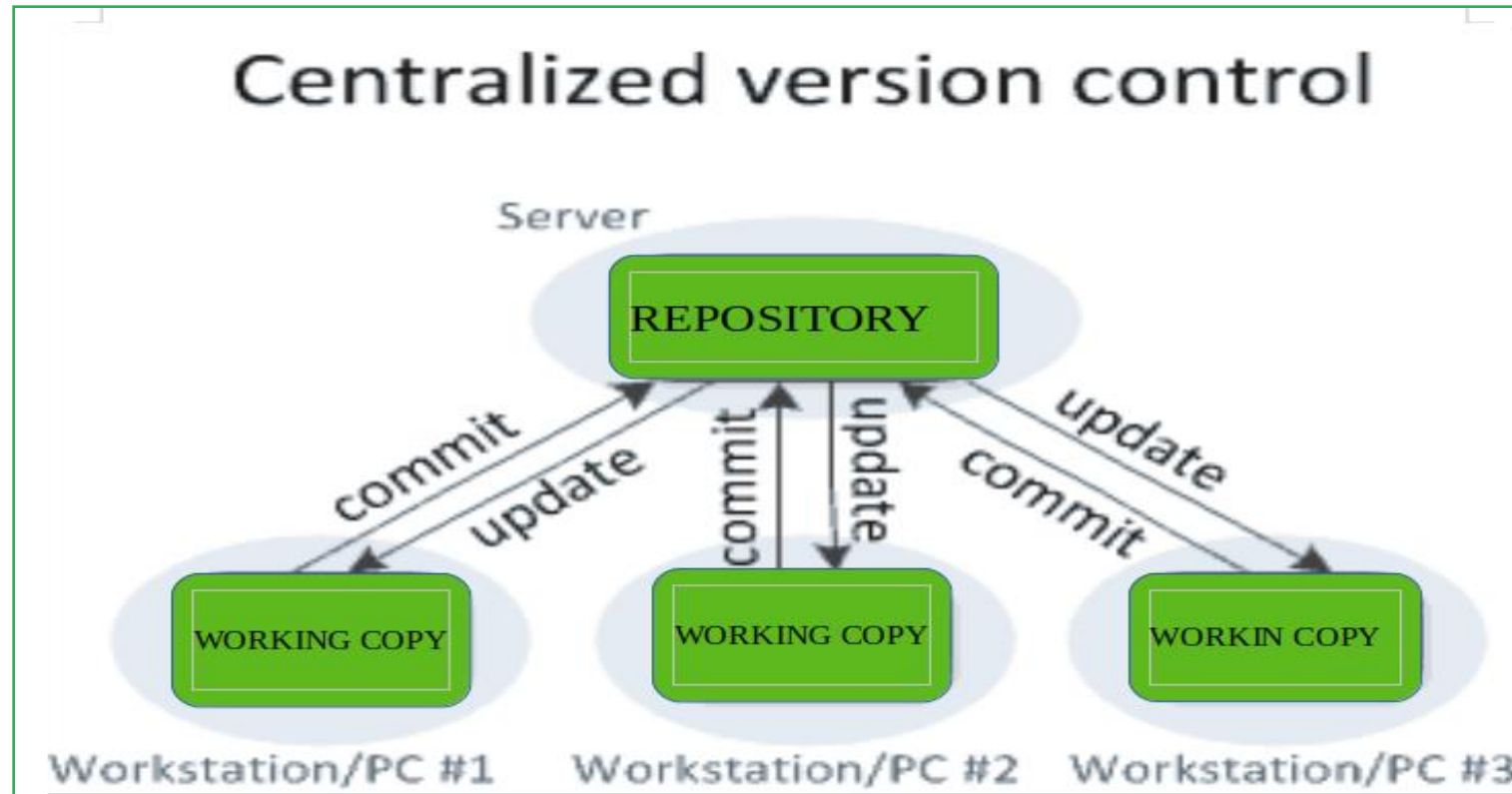


# Centralized Version Control:

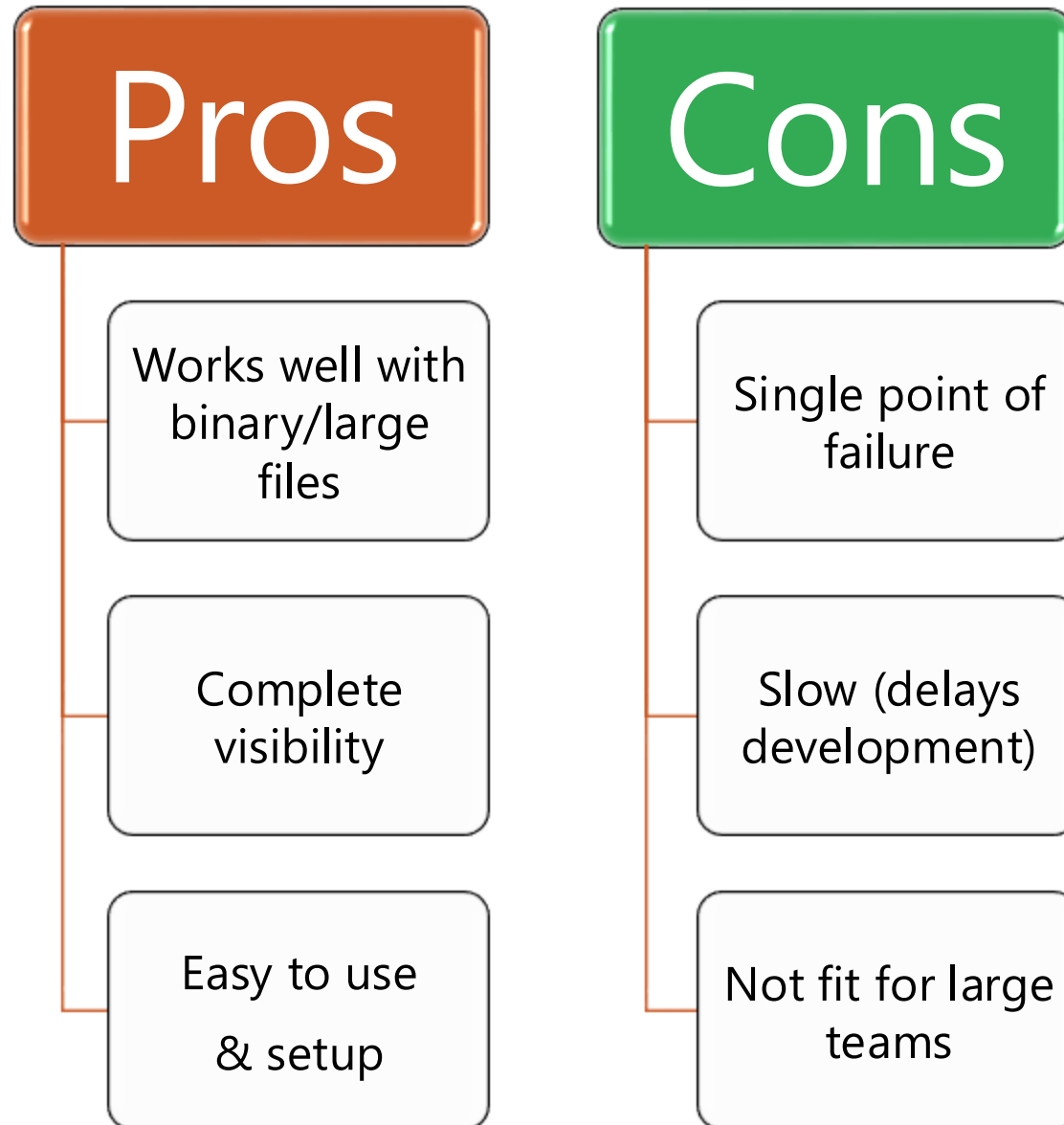
- A centralized version control system offers software development teams a way to collaborate using a central server
- Some common centralized version control SVN, TFvC, etc.

Action to perform to make changes visible:

- ☐ You Commit
- ☐ They Update



# Centralized Version Control:

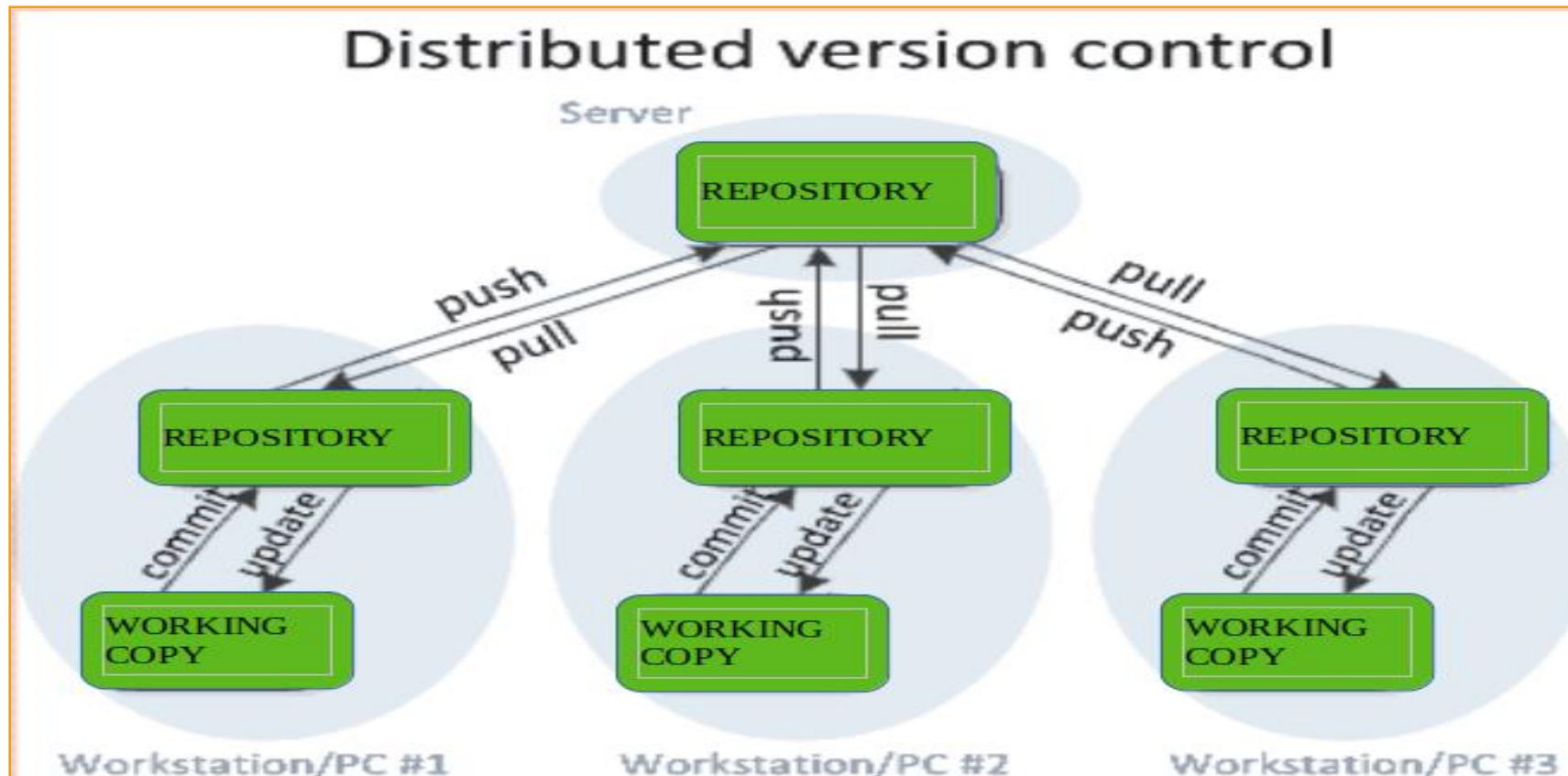


# Distributed Version Control:

- Distributed version control systems contain multiple repositories
- Each user has their repository and working copy, users need to pull to fetch the latest changes whereas developers need to push their changes to update the remote repository
- Some common centralized version control GIT, etc.

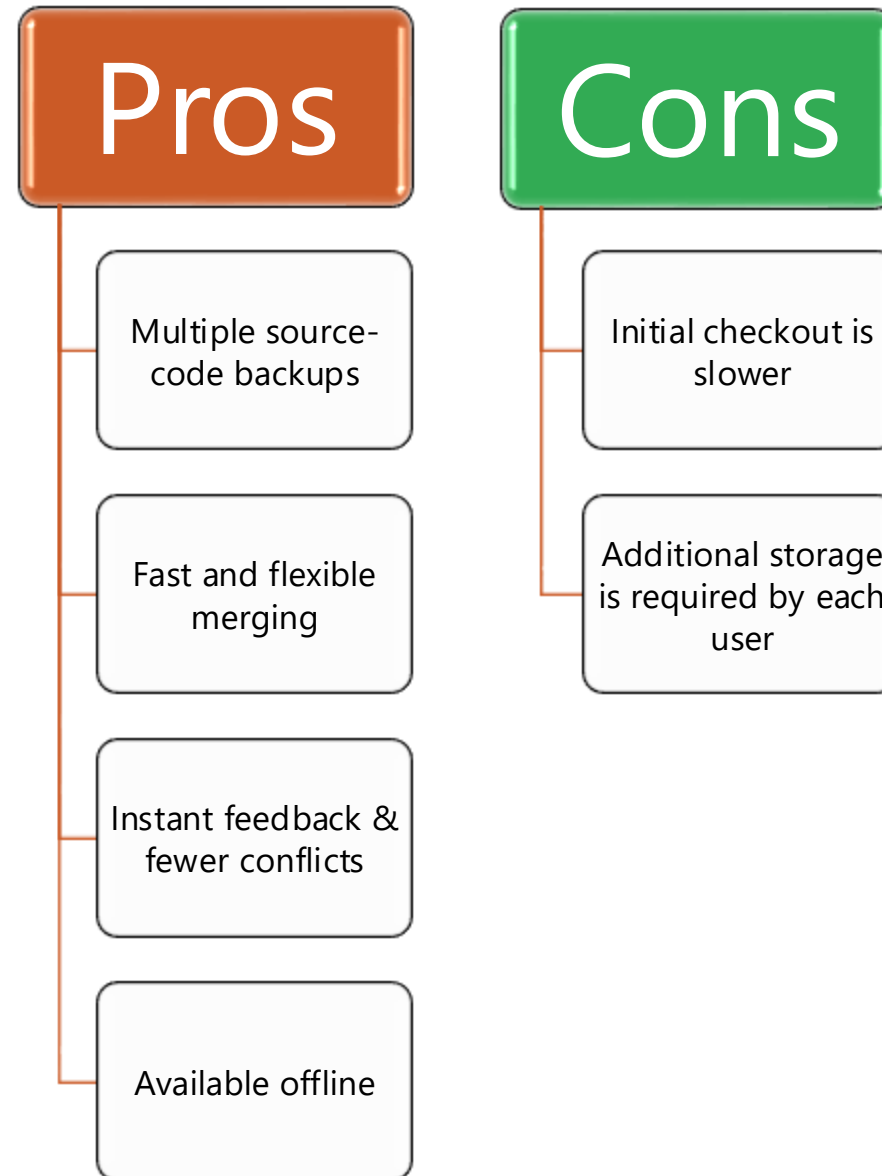
Action to perform to make changes visible:

- ☐ You commit
- ☐ You push
- ☐ They pull
- ☐ They update



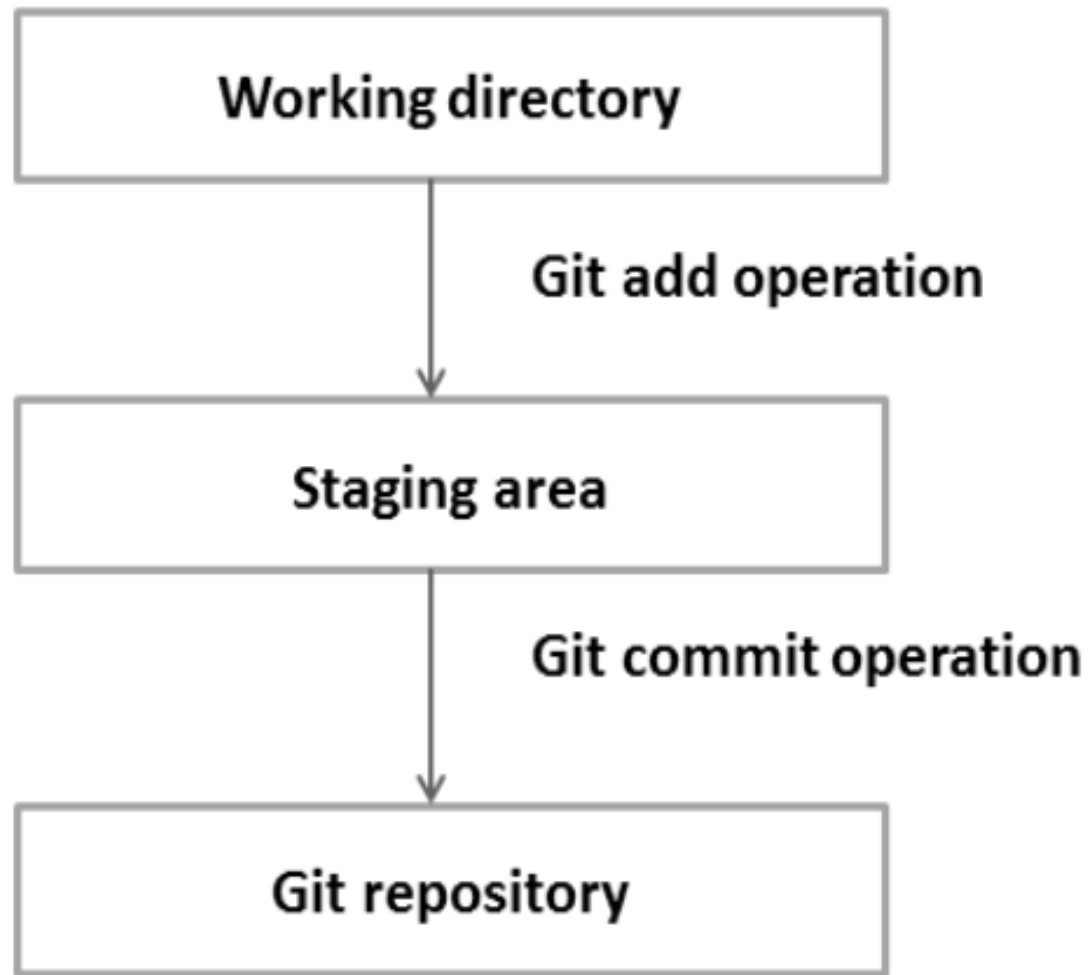


# Distributed Version Control:



**GIT** is a fast, scalable, distributed revision control system with an unusually rich command set that provides both high-level operations and full access to the internals

## GIT - Stages



# DEMO



# Summary:

- Introduction to Version Control System

Types of VCS:

- ❑ Centralized
- ❑ Distributed



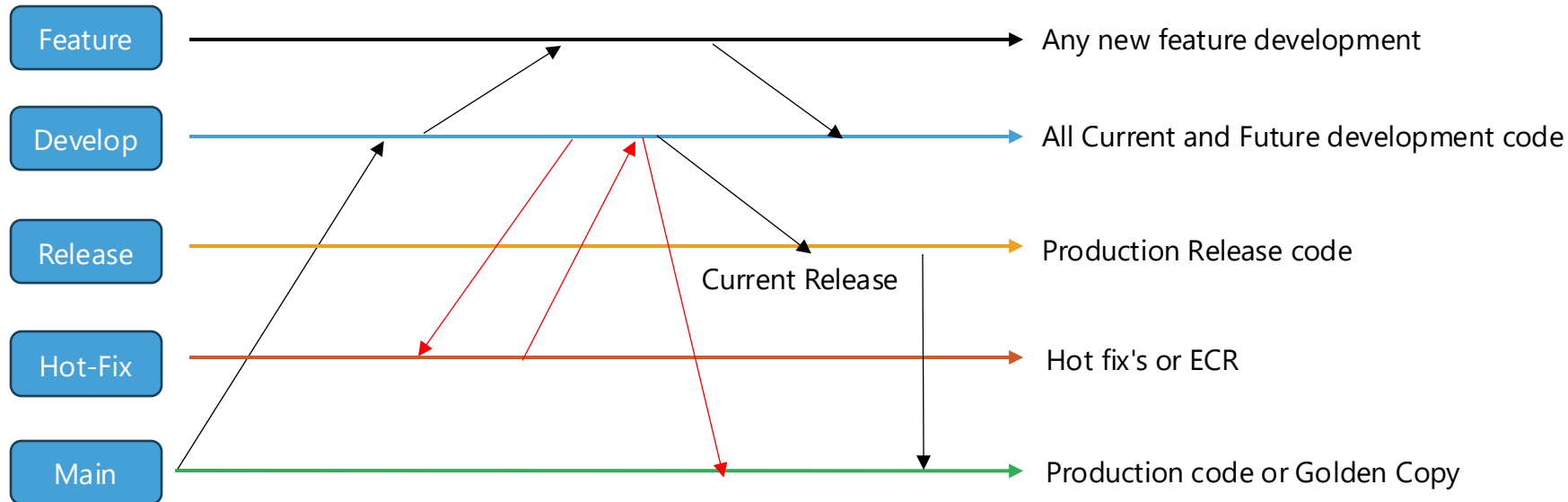
# Branching Strategies:

- Effective branch management is crucial for successful collaboration and efficient development with Git

Git-Flow is a comprehensive branching strategy that aims to cover various scenarios  
It defines specific branch responsibilities, such as

## ❑ Git-Flow:

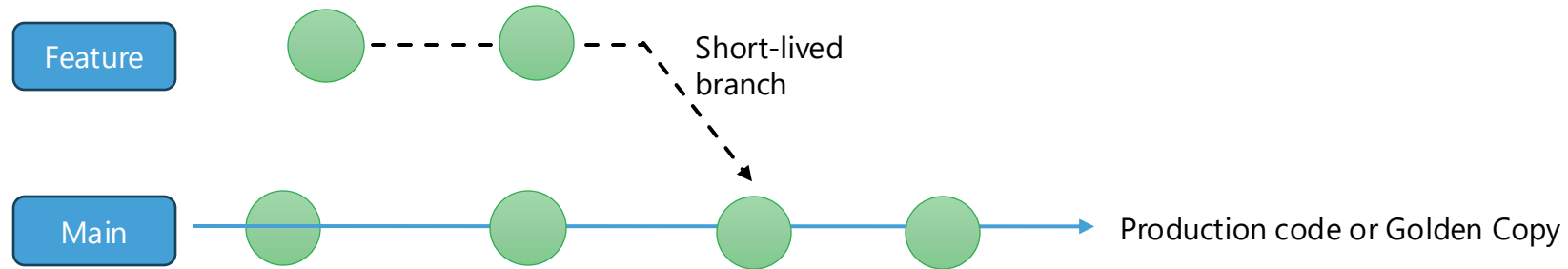
- main/master for production
- develop for active development
- feature for new features
- release as a gatekeeper to production
- hotfix for addressing urgent issues



# Branching Strategies:

## ❑ Trunk-based:

Trunk-based development is a version control management practice where developers merge small, frequent updates to a core “trunk” or main branch



# Azure Repos:

- Azure Repos is a set of version control tools that you can use to manage your code
- Azure Repos provides two types of version control:



GIT



TFVC



# Merge Types:

## ❑ Merge (no fast-forward):

- It is a default integration strategy in Azure Repos, GitHub, and most other Git providers
- All individual commits are preserved as-is, and a new merge commit is created
- This strategy is helpful because it illustrates exactly how a developer (or developers) worked on a pull request, including each individual, commit along the way



## ❑ Squash Commit:

- In Squash, each pull request becomes a single commit in the master, and there are no merges, just a simple, straight, linear history
- Individual commits are lost, which is best for teams that use “fix-up” commits or do not carefully craft individual commits for review before pushing them

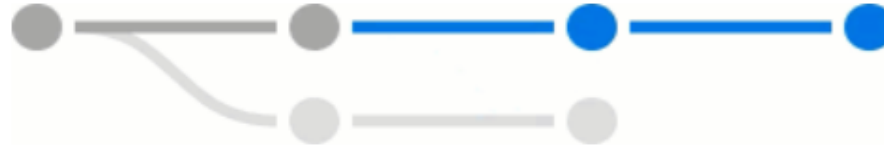




# Merge Types:

## ❑ Rebase:

- Rebase will take each commit in the pull request and cherry-pick them onto the destination branch
- When this strategy is used, history is straight and linear, like it is with the “squash” option, but each commit is retained



## ❑ Semi-linear merge:

- It's a mix of rebasing and a merge
- First, the commits in the pull request are rebased on top of the master branch. Then those rebased pull requests are merged into the destination branch
- This strategy is best used for retaining individual commits, and for viewing how the work evolved, but instead of just being rebased, a “merge bubble” is shown so that you can immediately see the work in each pull request



# Semantic Versioning:

- Semver is short for semantic versioning
- Semantic Versioning is a standardized way to give meaning to your software releases
- It is a universal way of versioning software development projects and tracking the variations

SemVer is in the form of a **Major.Minor.Patch**

Code status	Stage	Rule	Example version
First release	New product	Start with 1.0.0	1.0.0
Backward compatible bug fixes	Patch release	Increment the third digit	1.0.1
Backward compatible new features	Minor release	Increment the middle digit and reset last digit to zero	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit and reset middle and last digits to zero	2.0.0



# DEMO



# Summary:

- Introduction to Azure Repos

- Types of Merge
- Cherry-picking concept
- Branch Policies
- Branch Security
- Tags



# Talisman

By: Rohit K Singh



# Agenda:

- Type of Pre-commits / Pre-push hooks
- Introduction to Talisman
- Talisman Installation
- Demo




# Pre-commit / Pre-push hooks:

- Sensitive information such as the **access keys, access tokens, SSH keys, etc.** are often erroneously leaked due to accidental git commits
- Pre-commit hooks can be installed on developers' workstations to avoid them
- Work on a pure Regex-based approach for filtering sensitive data
- If developers want they can bypass this step

## Dev put AWS keys on Github. Then BAD THINGS happened

Fertile fields for Bitcoin yields - with a nasty financial sting

 [Darren Pauli](#)

Tue 6 Jan 2015 // 13:02 UTC

Bots are crawling all over GitHub seeking secret keys, a developer served with a \$2,375 Bitcoin mining bill found.

DevFactor founder Andrew Hoffman said he used [Figaro](#) to secure Rails apps which published his Amazon S3 keys to his GitHub account.

He noticed the blunder and pulled the keys within five minutes, but that was enough for a bot to pounce and spin up instances for Bitcoin mining.

"When I woke up the next morning, I had four emails and a missed phone call from Amazon AWS - something about 140 servers running on my AWS account," Hoffman

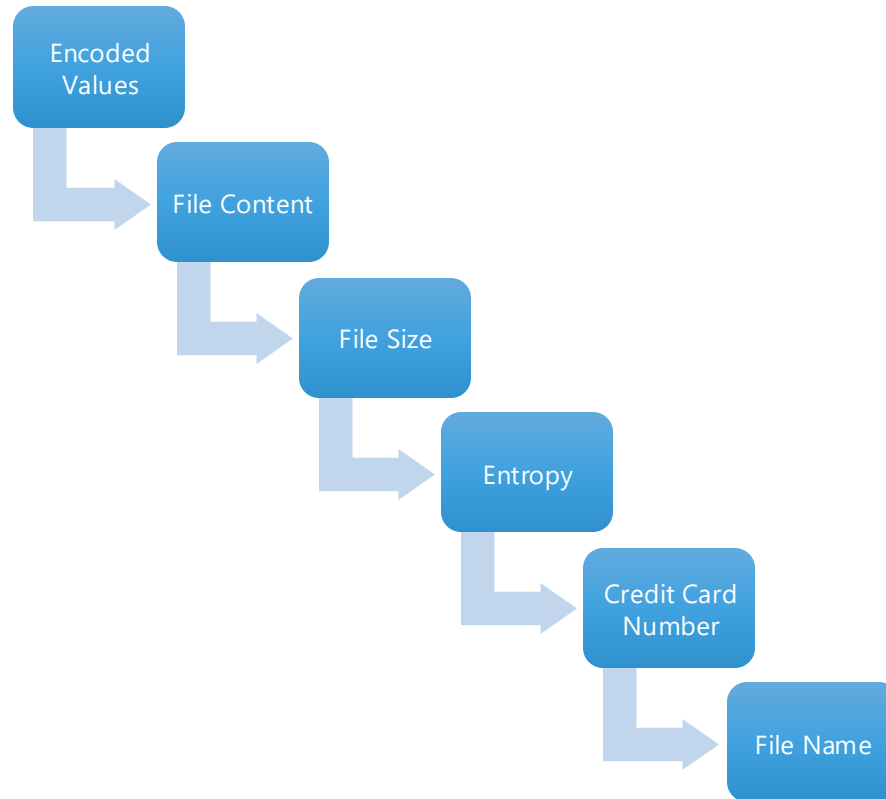
[AWS Keys - Case Study](#)



# Talisman:

- Talisman installs a hook to your repository to ensure that potential secrets or sensitive information do not leave the developer's workstation
- It validates the outgoing change for things that look suspicious like potential SSH keys, authorization tokens, private keys, etc.

Talisman works on pattern matching:





# Installation:

## Single Project Installation

Talisman will be present only in a single git repository

Pre-push hooks:

```
➤ curl https://thoughtworks.github.io/talisman/install.sh > ~/install-talisman.sh
```

```
➤ chmod +x ~/install-talisman.sh
```

```
➤ cd my-git-project
```

```
➤ ~/install-talisman.sh
```

```
Downloading talisman_linux_amd64 from https://github.com/thoughtworks/talisman/releases/download/v1.11.0/talisman_linux_amd64
Downloading checksums from https://github.com/thoughtworks/talisman/releases/download/v1.11.0/checksums
talisman_linux_amd64: OK
Talisman successfully installed to '.git/hooks/pre-push'.
```

## Global Installation:

Talisman will thus be present, not only in your existing git repositories but also in any new repository that you 'init' or 'clone'

Pre-hooks:

```
➤ curl --silent https://raw.githubusercontent.com/thoughtworks/talisman/master/global_install_scripts/install.bash > /tmp/install_talisman.bash
  && /bin/bash /tmp/install_talisman.bash
```

Post-hooks:

```
➤ curl --silent https://raw.githubusercontent.com/thoughtworks/talisman/master/global_install_scripts/install.bash > /tmp/install_talisman.bash
  && /bin/bash /tmp/install_talisman.bash pre-push
```



# DEMO



# Summary:

- Avoid pushing sensitive information into the Cloud

- Understanding the concept of Talisman and Integration with existing or new repositories



# Azure AD Service Connection Agent Pools

By: Rohit K Singh



# Agenda:

- Introduction to Azure AD
- Authentication and Authorization flow
- Service Principle
- Managed Identity
- Service Connection – Concept
- Introduction to Azure Agent Pools
  - Self-hosted
  - Microsoft-hosted
- Demo



# Azure Active Directory (Entra ID)

- Azure Active Directory is Microsoft's multi-tenant, cloud-based directory and identity management service
- Azure AD helps employees sign up for multiple services and access them anywhere over the cloud with a single set of login credentials

3 main users of Azure AD:



IT Admins

- Manage role permissions
- Control access



App Developers

- single sign-on
- API Interfaces

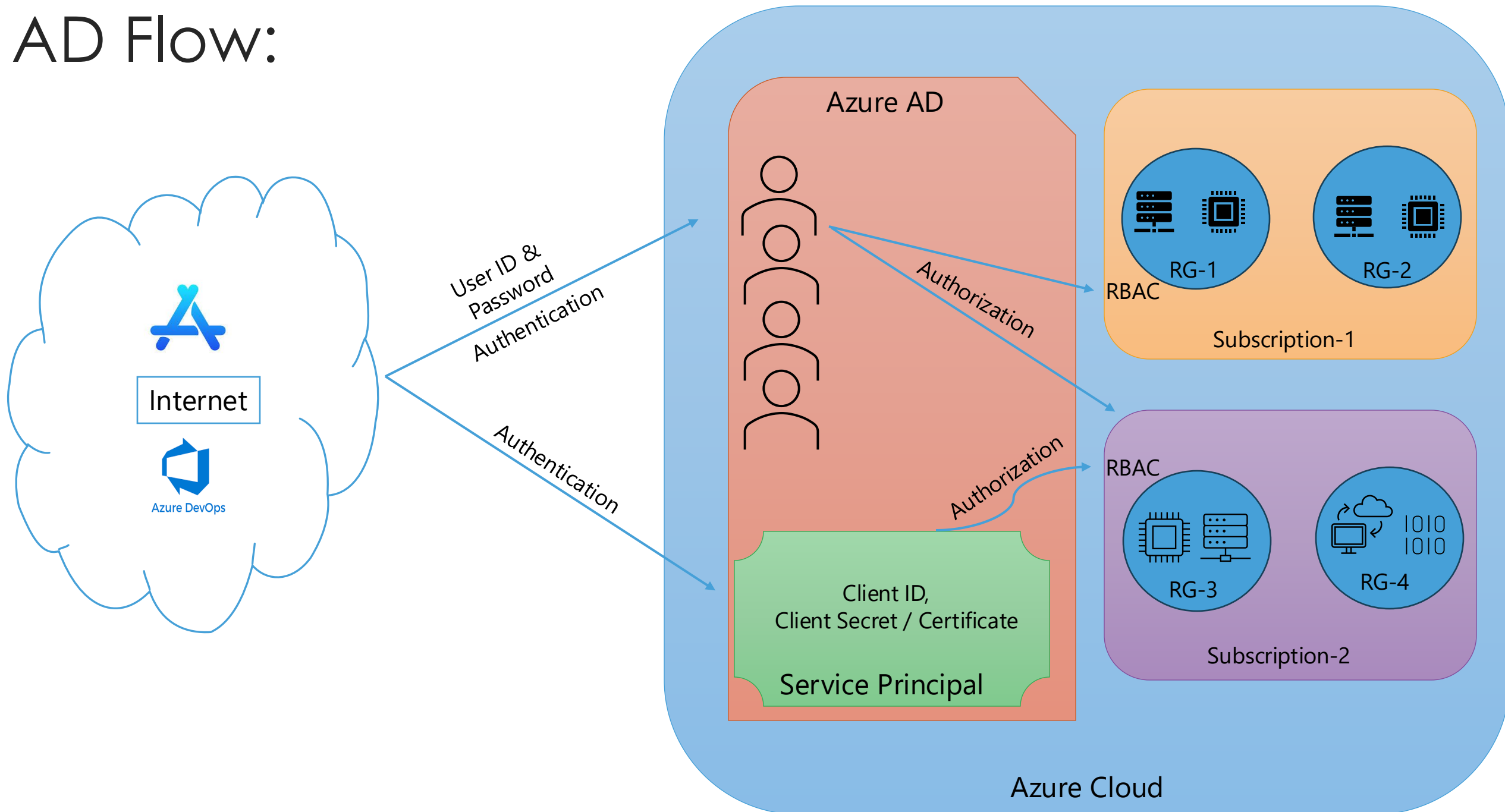


Cloud service subscribers

- Office 365
- Azure cloud services



# AD Flow:



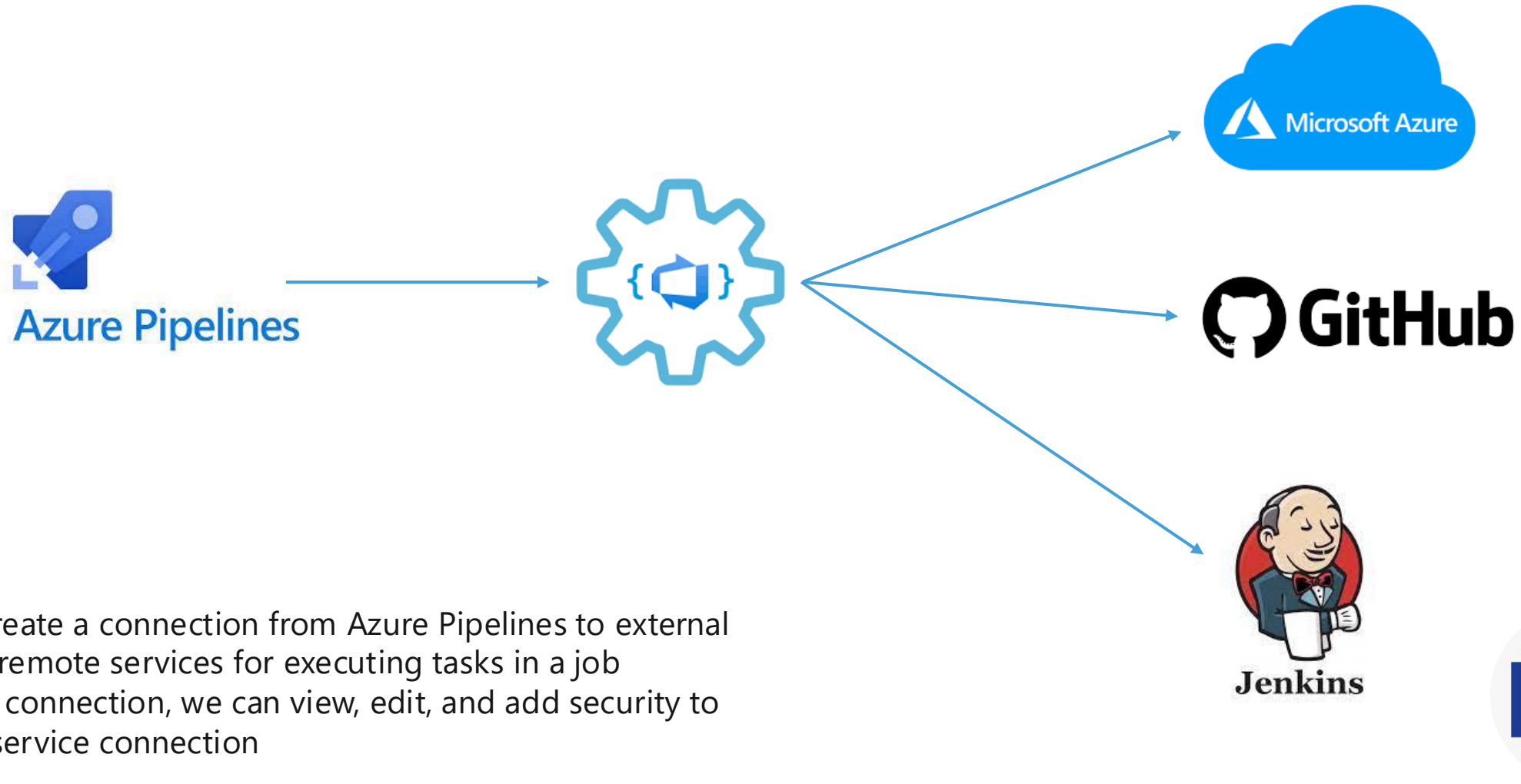
# Service Principal Vs Managed Identity:

	Managed Identity	Service Principal
Creation	Automatically created and managed by Azure	Must be manually created and managed by the user
Lifecycle	Tied to the lifecycle of the resource it's assigned to. When the resource is deleted, the identity is also deleted	Independent of any resource and must be manually deleted
Permissions	Permissions are directly assigned to the resource	Permissions are assigned to the service principal, which can be used across multiple resources
Rotation of Secrets	No need to manage secrets as Azure takes care of it	User is responsible for managing and rotating secrets
Usage	Can only be used within the Azure environment	Can be used both within and outside of Azure
Scope	Limited to the resource it's assigned to	Can be used across multiple resources and services







# Service Connection:



# Agent Pools:

- An agent is computing infrastructure with an installed agent software that runs one job at a time
- To build your code or deploy your software using Azure Pipelines, you need at least one agent
- When your pipeline runs, the system begins one or more jobs

There are 2 main types of agent pool:

- Microsoft-hosted agents  Agents hosted and managed by Microsoft
- Self-hosted agents  Agents that you configure and manage, hosted on your VMs



DEMO



# Classic Pipelines

By: Rohit K Singh

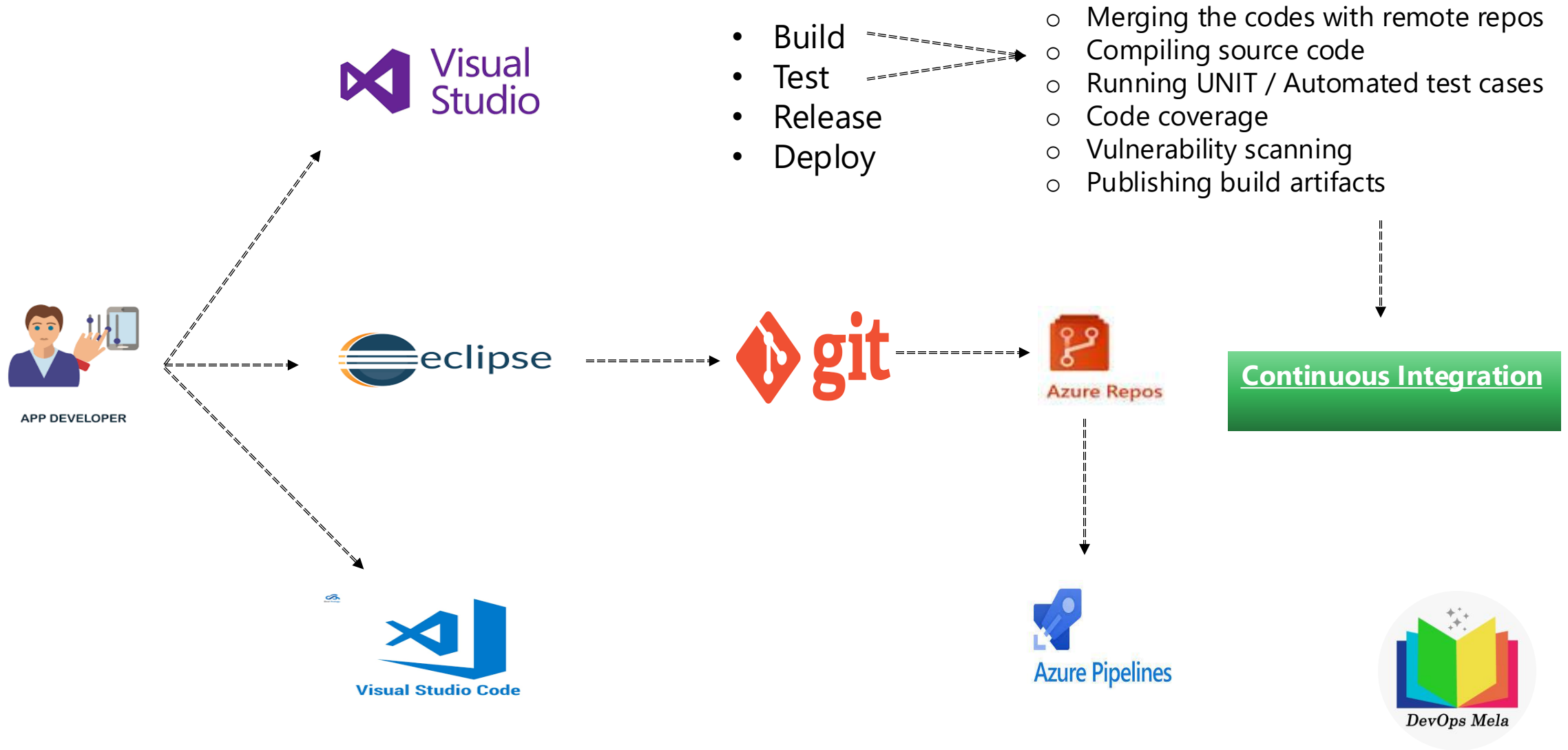


# Agenda:

- Continuous Integration - Introduction
- Continuous Delivery & Deployment – Introduction
- Monolithic vs Microservice Apps
- ASP.NET Application: Use case
- Create 1<sup>st</sup> Classic build asp.net pipeline – Demo
  - Types of Build Triggers
- Create 1<sup>st</sup> Classic release pipeline – Demo
  - Pre-Post release condition
  - Pipeline variables
- Azure App Service - Introduction
- Deployment Groups
- Task Groups

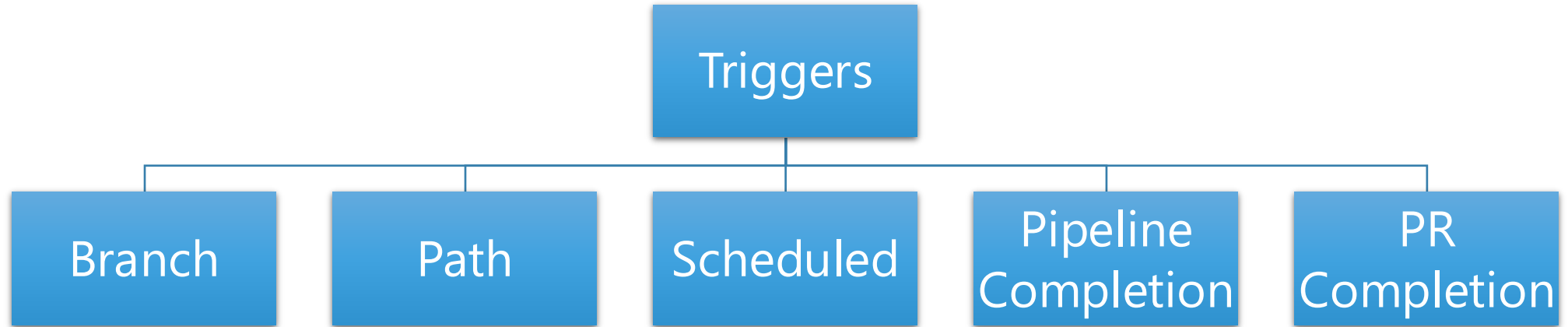


# Continuous Integration:

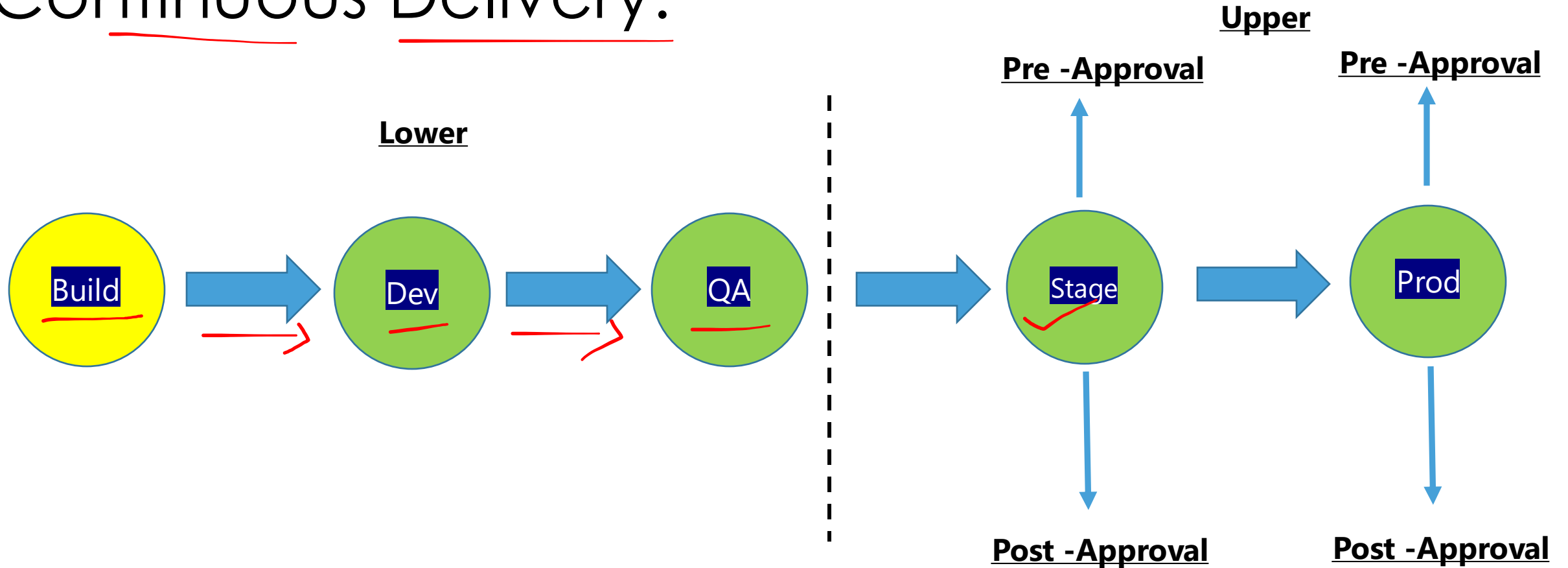


# Build Triggers:

- Build Triggers trigger the build of the project for continuous integration and continuous deployment



# Continuous Delivery:

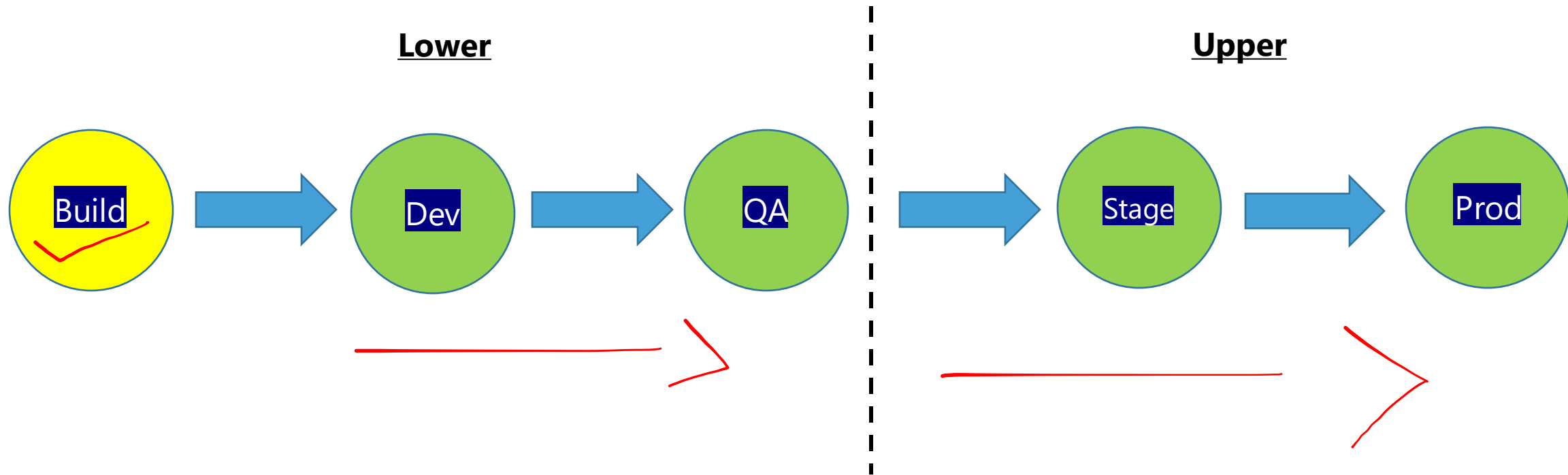


- Continuous delivery is an automated release process, we can deploy our application at any time by clicking a button
- With continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements
- With continuous delivery, codes get deployed in lower environments without manual intervention but for upper environments, deployment manual approval is needed

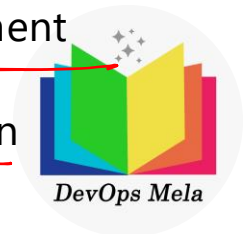




# Continuous Deployment:



- Continuous deployment goes one step further than continuous delivery
- With continuous deployment, every change that passes all stages gets deployed till the production environment without human intervene
- With continuous deployment, only a failed test will prevent a new change from being deployed to production
- With continuous deployment, we can completely bypass release day and accelerate the feedback loop with customers



# Monolithic Applications:

Large applications that have many different components all packaged up into one executable over time

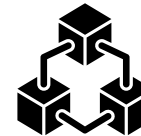
Slow to build, test, and deploy



Hard to implement changes because the code is tightly coupled



Difficult to test individual components



The entire application needs to be deployed every time there is an update



# Microservice Applications:

Loosely coupled services running as independent components on various resources

## Advantage

### Reduced Coupling

A change in one component is less likely to cause an issue with another component

### Agility

Easier to build, test, and deploy because you can focus on each component individually

### Scalability

Can scale components separately

## Disadvantage

### Complexity

Harder to keep track of where everything is running from

### Latency

All communication between components is through network calls



# Azure App Service:

- Azure App Service is an HTTP-based service for hosting web applications, REST APIs, and mobile backends
- Azure App Service is a fully managed platform as a service (PaaS) offering for developers
- It supports various languages such as .NET, .NET Core, Java, Node.js, PHP, and Python
- Applications run and scale with ease on both Windows and Linux-based environments
- App Service can be very easily integrated with DevOps tools like Azure DevOps, Github, etc.

Key features of App Service:

**Multiple languages and frameworks**

**Managed production environment**

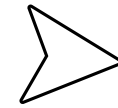
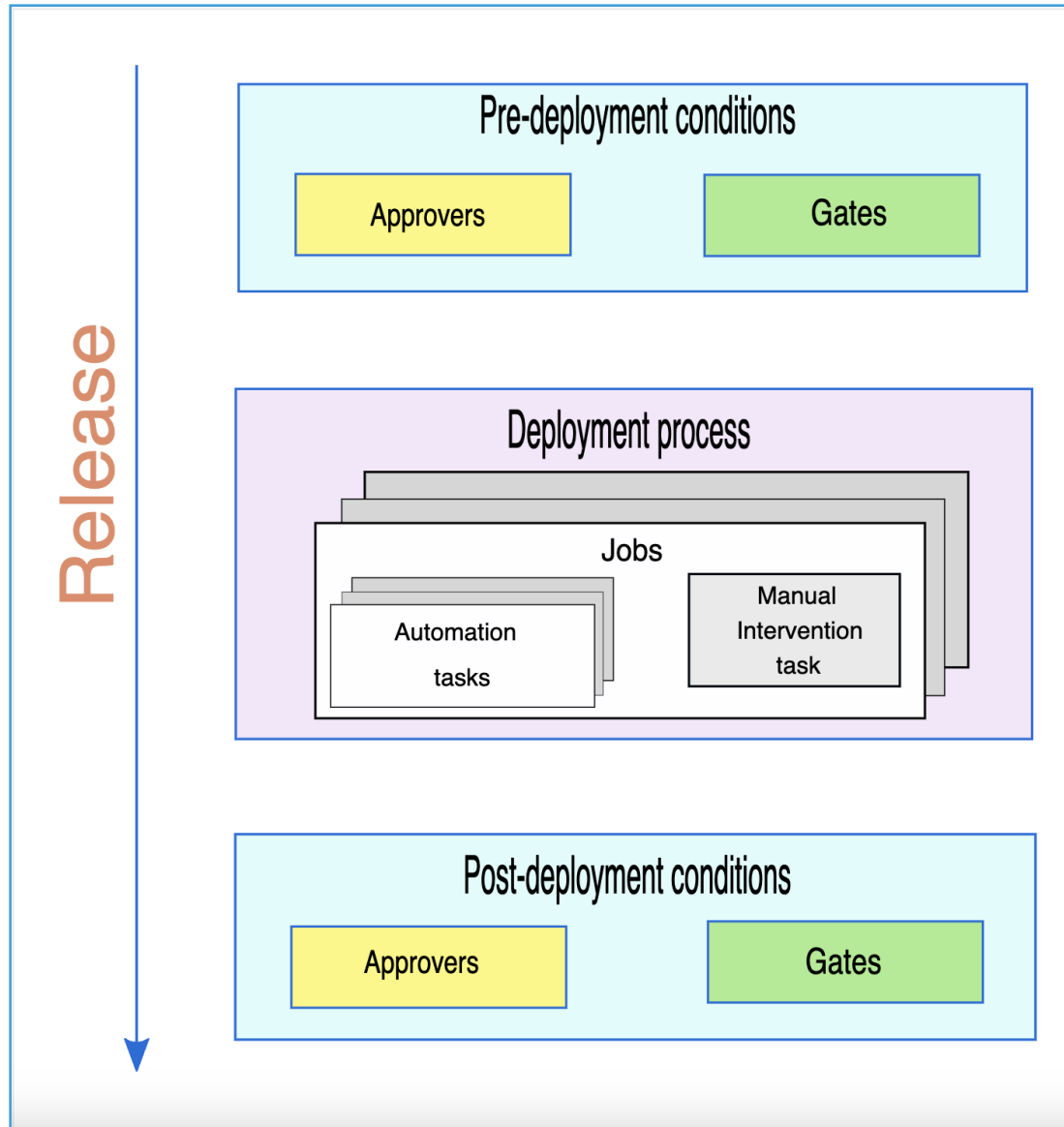
**Containerization and Docker**

**DevOps optimization**

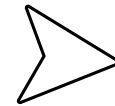
**Global scale with high availability**



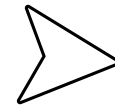
# Pre-Post Condition:



Using the Approvals and Gates feature to control the workflow of the deployment pipeline



Each stage can be configured with pre-post conditions like manual approval, checking for specific conditions, etc.



Pause the deployment pipeline flow and carry out any manual tasks, health checks, etc.



# Pipeline Variables:

- Variables give you a convenient way to get key bits of data into various parts of the pipeline
- The most common use of variables is to define a value that you can then use in your pipeline
- Variables are strings and are mutable

## Ways to pass variables in pipelines:

### ➤ Pipeline variables:

- Local to respective pipeline
- Values can be passed and used in the pipeline tasks
- Environmental scope can be set at the variable level
- Variables values can be kept secret

### ➤ Variable groups:

- Stores values and secrets values
- These values can be shared across multiple build and release pipelines

**Syntax:** \$(variable\_name)

Name	Value
resourcegroup	devopsmela-rg-dev
resourcegroup	devopsmela-rg-qa
serviceplan	service-plan-dev
serviceplan	service-plan-qa
webapp	devopsmela-dev123
webapp	devopsmela-qa

### Library

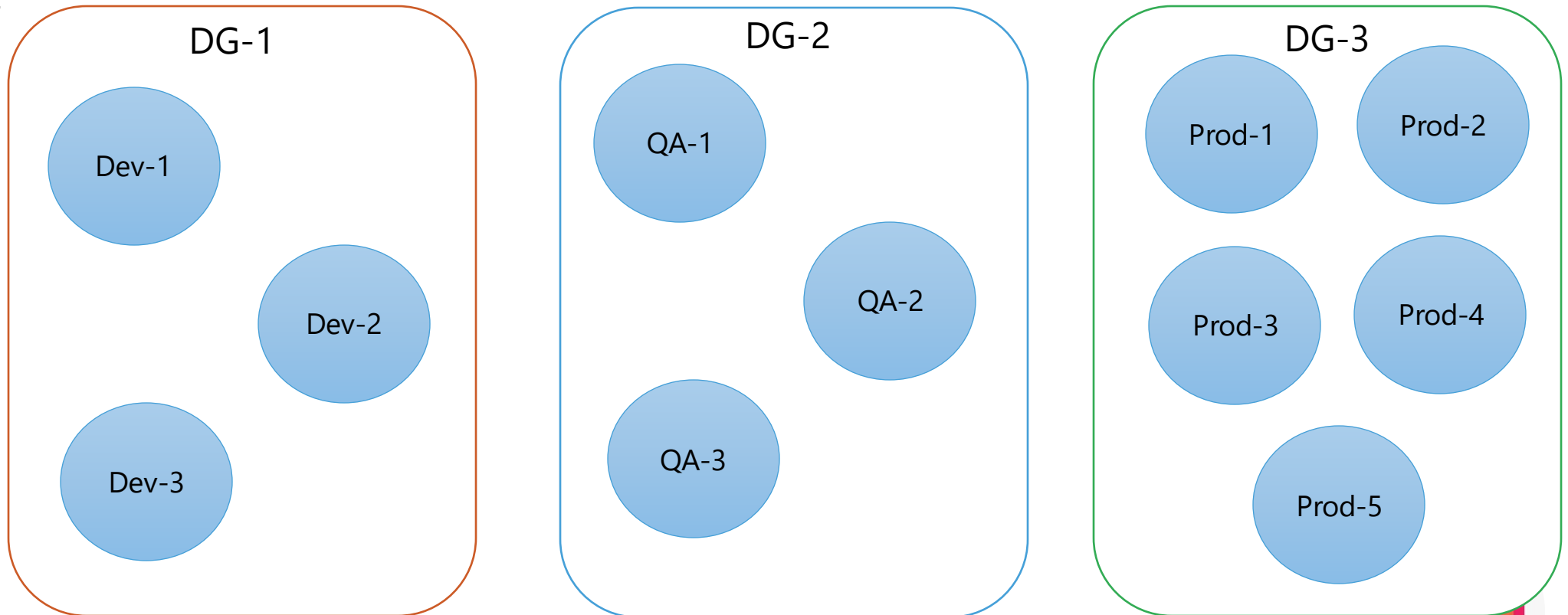
Variable groups   Secure files   + Variable group   Security

Name ↕	Date modified
fx sqldb	11/29/2023
fx TF_Variables	11/17/2023



# Deployment Groups:

- A deployment group is a logical set of deployment target machines that have agents installed on each one
- Deployment groups represent the physical environments; for example, "Dev", "Test", or "Production" environment




**\*\*Deployment groups are only available with Classic release pipelines\*\***

# Task Groups:

- Encapsulate a sequence of tasks
- Into a single reusable task, can be integrated with Build & Release pipelines
- Task groups can be easily added to the JOBS from the task catalogs
- Use task groups to standardize and centrally manage the deployment step
- The changes in the task group will automatically elsewhere

 Task groups |  Import  Security

 Filter by name



Name ↑	Modified by	Description	Date modified
TF_Task	meladevops	Terraform Infra Task	11/17/2023

## Note

Task groups are not supported in YAML pipelines. Instead, in that case you can use templates.  
See [YAML schema reference](#).





DEMO



# Azure Artifacts

By: Rohit K Singh




















# Agenda:

- Package Management - Introduction
- Azure Artifacts – Introduction
- Creating 1<sup>st</sup> Azure Artifacts feeds
- Demo



# Package Management:

- Package managers, sometimes referred to as package management systems, are essential tools
- It allows users to:
  - Install
  - Upgrade
  - Configure
  - Manage software packages
- They allow developers to easily add, update, or remove packages from their projects

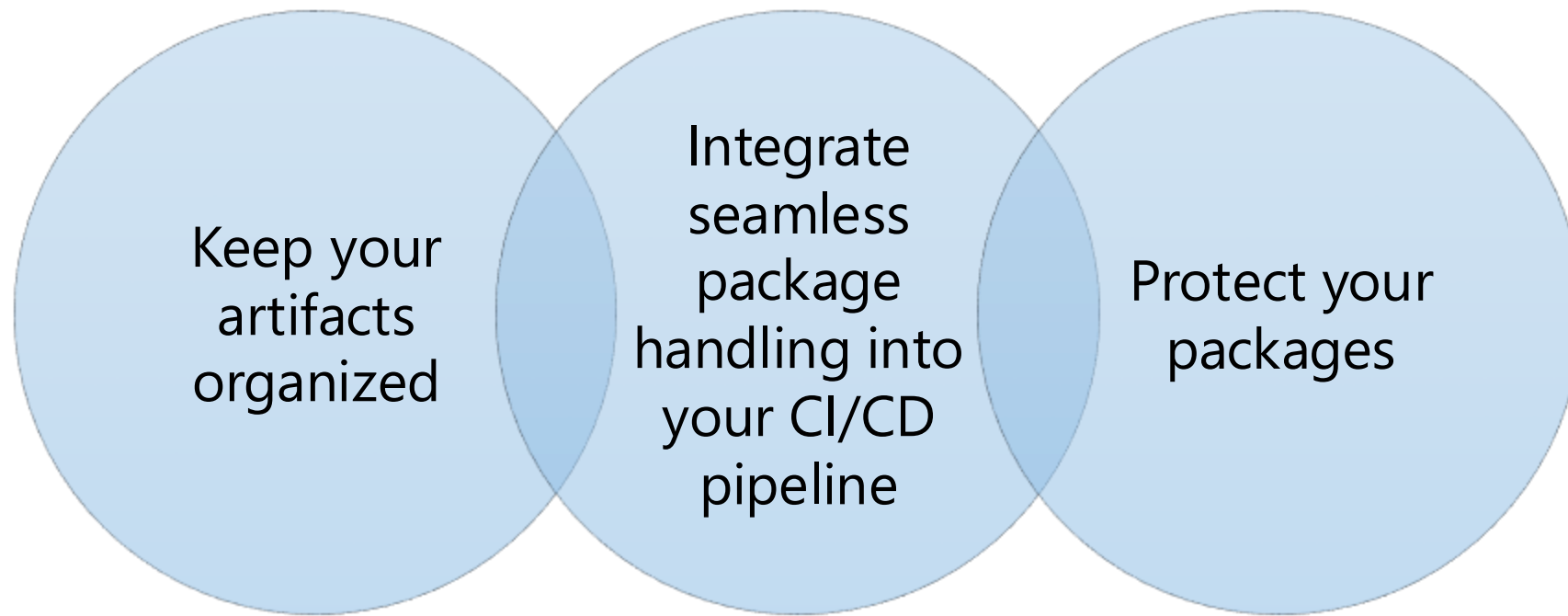
	Npm	▼		NuGet	▼		Chocolatey	▼
	Composer	▼		Ninite	▼		Pip	▼
	APT	▼		Aptitude	▼		Docker	▼
	Homebrew	▼		Apache Maven	▼		RPM package Manager	▼
	Rubygems	▼		Yarn	▼		Dpkg	▼
	Cargo	▼		Pacman	▼			



# Azure Artifacts:

- Azure Artifacts is an extension that makes it easy to discover, install, and publish NuGet, NPM, and Maven packages in Azure DevOps
- It's deeply integrated with other hubs like build so that package management can become a seamless part of your existing workflows
  - Using Azure Artifacts one can create, host, and share packages with teams
  - We can share code across teams, and manage all package types like Maven, NPM, Gradle, NuGet, etc

## Benefits:



# Maven

By: Rohit K Singh



# Agenda:

- Maven - Introduction
- Understanding POM Concept
- Maven lifecycle phases
- Maven setup and running through lifecycle phases - Demo



# Maven:

- Maven, a **Yiddish word** meaning accumulator of knowledge, was built as an attempt to simplify the build processes
- Maven is a powerful project management tool that is based on **POM** (project object model)
- It is used for project build, dependency, and documentation



## Helps Us with:

- Builds Features migration
- Documentation
- Reporting
- SCMs
- Releases
- Distribution

## Simplify build process

## Uniform build process

(maven project can be shared by all the maven projects)

## Provides project Information

(log document, cross-referenced sources, mailing list, dependency list, unit test reports etc.)





# POM (project object model)

- **POM.xml** files contain project and configuration information for Maven to build the project
- The POM files have information such as project dependencies, source directory, and test source directory. plugin, goals, etc

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xsi:schemaLocation="http://maven.apache.org/POM
<modelVersion>4.0.0</modelVersion>
<groupId>com.mycompany.app</groupId>
<artifactId>my-app</artifactId>
<packaging>jar</packaging>
<version>1.0-SNAPSHOT</version>
<name>my-app</name>
<url>http://maven.apache.org</url>
```

— — — ➔ Root element of pom.xml file

— — — — — ➔ It specifies the model version

— — — — — ➔ It specifies project group ID

— — — — — ➔ It specifies project artifacts ID

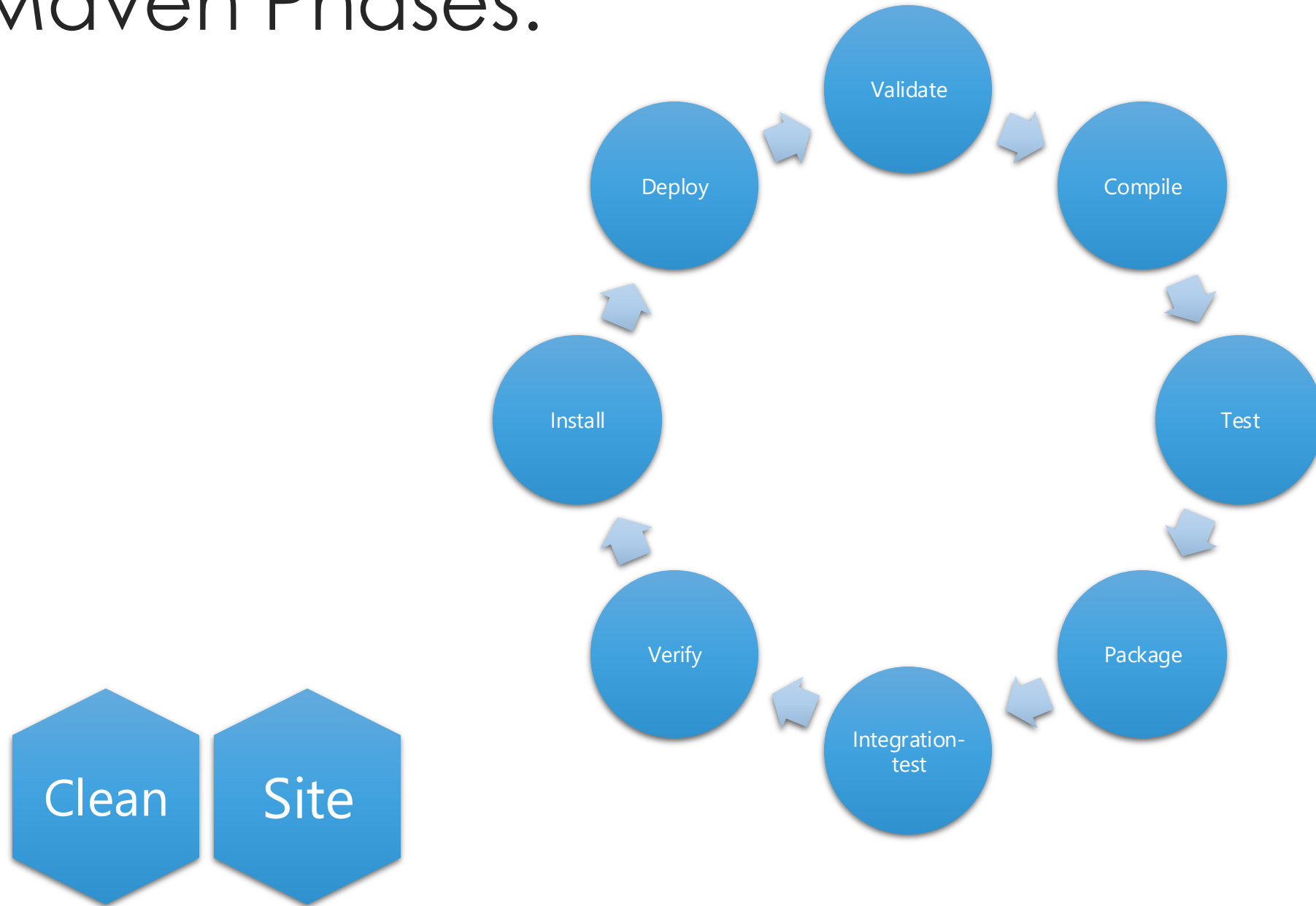
— — — — — ➔ Defines packaging types such as jar, war, etc.

— — — — — ➔ Artifacts version

— — — — — ➔ Defines the name of the maven project



# Maven Phases:



DEMO



# YAML basic's

By: Rohit K Singh



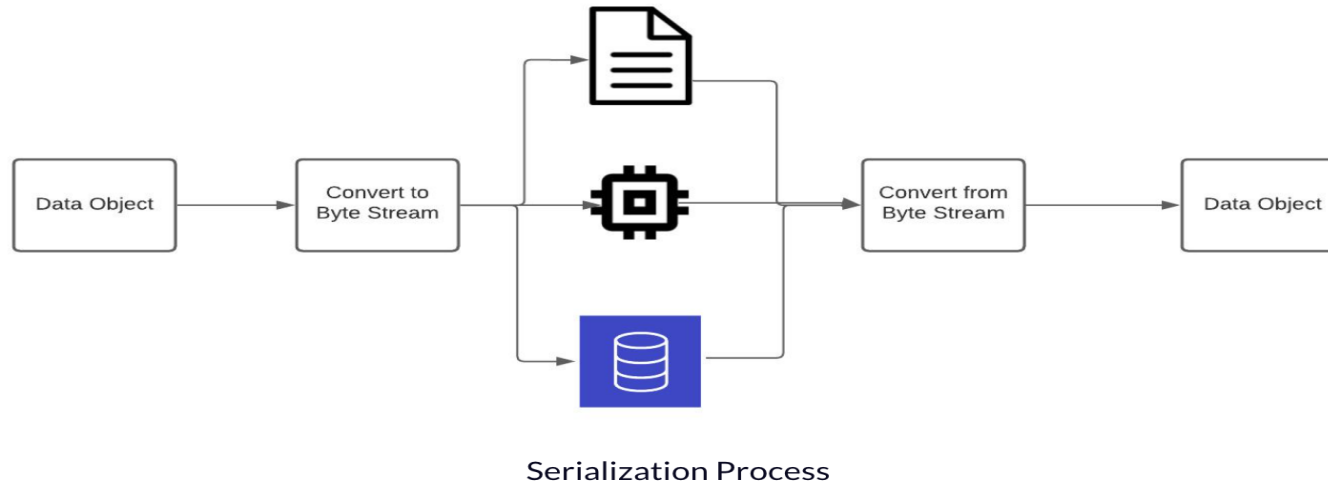
# Agenda:

- Data Serialization - Introduction
- Understanding YAML
- Basic YAML Syntax
- DEMO



# Data Serialization:

Data serialization is the process of converting data objects present in complex data structures into a byte stream for storage, transfer, and distribution purposes on a physical device



**Example:** Serialization is when reading data from databases and transferring it across the web

Some commonly used serialization formats are JSON, YAML, XML



# YAML:

- YAML is a data serialization format that stands for YAML ain't Markup language
- YAML is written in simple English and is easier for humans to read and write
- YAML supports various data types like cases, arrays, dictionaries, lists, and scalars
- It has good support for the most popular languages like JavaScript, Python, Ruby, Java, etc
- YAML only supports spaces, and it is case-sensitive as well as space-sensitive
- Tabs are not accepted universally. A YAML file has a .yaml extension

```
1 trigger:
2   - main
3
4 pool:
5   - vmImage: ubuntu-latest
6
7 steps:
8
9   Settings
10  - task: SonarQubePrepare@5
11    inputs:
12      SonarQube: 'sonar-scan'
13      scannerMode: 'Other'
14      extraProperties: |
15        # Additional properties that will be passed to the scanner,
16        # Put one key=value per line, example:
17        # sonar.exclusions=**/*.bin
18        sonar.projectKey=Azure_DevSecOps_6_java_webapp_AYv511VbgeyYyTxZ50lo
19        sonar.projectName=java_webapp
20        sonar.qualitygate.wait=true
21        sonar.qualitygate.timeout=300
```



# Basic Syntax:

## Mapping:

- The mapping syntax is **key: value**. (Note the space, it's very crucial in YAML, unlike JSON or XML)

---

```
name: DevOps Mela  
age: 32  
location: Mumbai
```





# Basic Syntax:

## Data Types:

- YAML also supports data types like characters, strings, integers, floating values, and collections like arrays, lists that are constructed from basic data types

---

NAME: "DevOps Mela"

MALE: FALSE

Amount: 120.30

Concern: NULL

AGE: 16

----->

String

----->

Boolean

----->

Float

----->

Null

----->

Integers



# Basic Syntax:

## List

```
---  
  
- apple  
  
- banana  
  
- mango
```

## Nested List

```
---  
  
student: "john"  
hobbies:  
  - music  
  - reading  
  - dancing
```

## Nested Dictionary

```
---  
  
student2:  
  fatherName: "Grey"  
  motherName: "Winey"  
  subjectDetails:  
    subject1: 70  
    subject2: 100
```



DEMO



# YAML Pipelines

By: Rohit K Singh

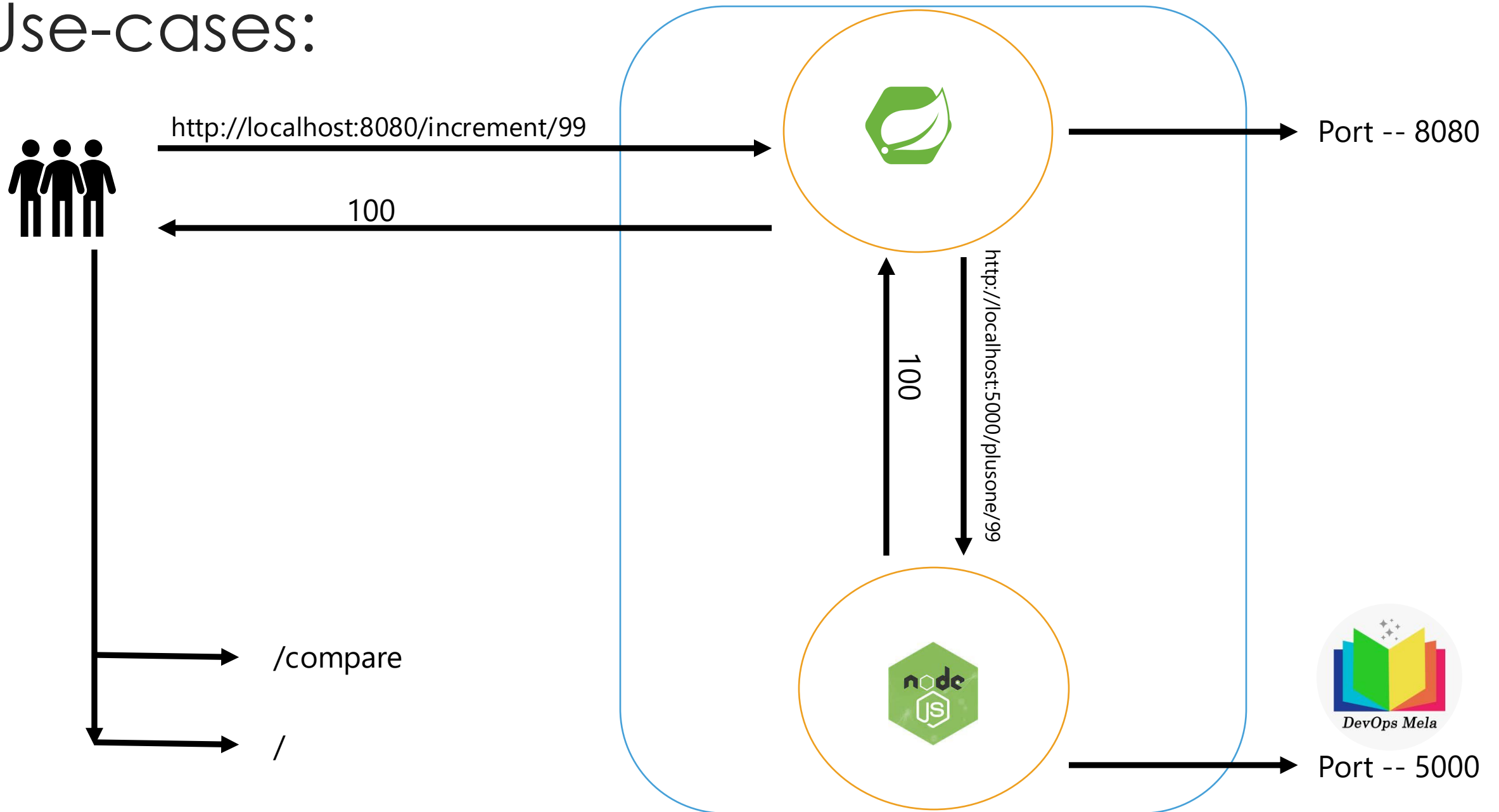


# Agenda:

- JAVA Project: Use-Case
- Create 1<sup>st</sup> YAML-based build Java Maven pipeline – Demo
- Integration of Test tools to build pipeline
  - Junit
  - Code Coverage
  - Static Code Analysis (SonarQube)
  - Vulnerability Scan (Mend Bolt - formerly WhiteSource)
- Deployment Strategies
  - Run-once
  - Rolling
  - Canary
- Create 1<sup>st</sup> YAML based release - Demo
- Environments



# Use-cases:



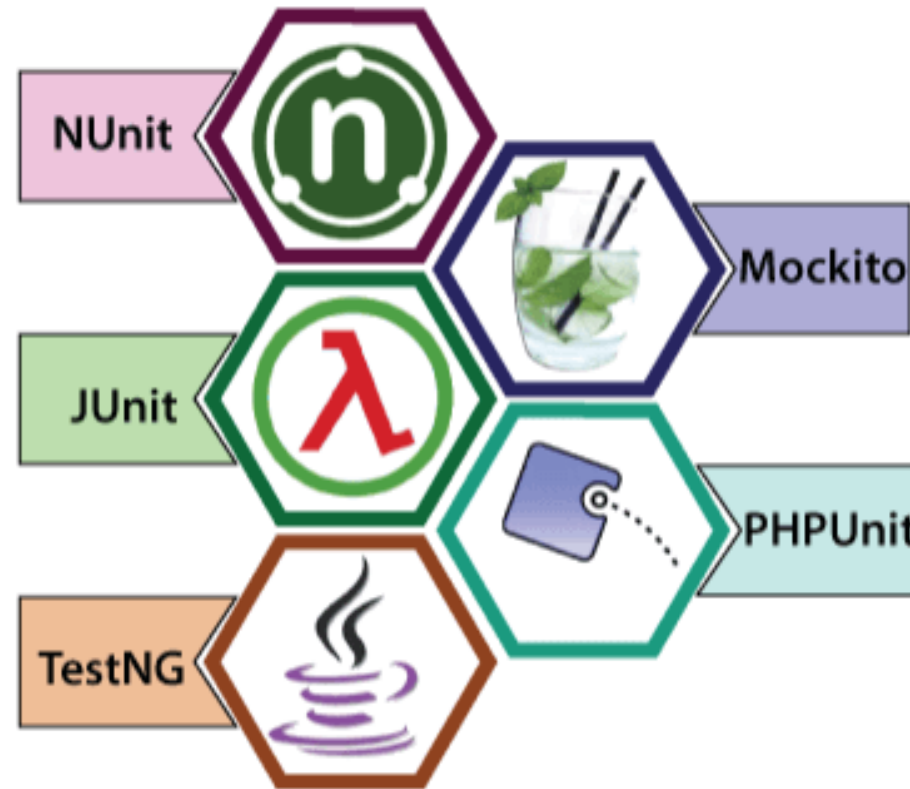
# Unit Test:

- A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system
- In most programming languages, that is
  - Function,
  - Subroutine,
  - Method,
  - Property

## JUnit Tool:

- JUnit is a unit-testing open-source framework for the Java programming language
- Java Developers use this framework to write and execute automated tests
- JUnit plays a huge role when it comes to regression testing

## Unit Testing Tools



# Code Coverage:

- Code coverage also called **test coverage**, is a percentage measure of the degree to which the source code of a program is executed when a particular test suite is run
- In other words, it **describes the percentage of code covered by automated tests**

Some of the most common metrics include:

- **Statement coverage:** % of statements in the code the tests execute
- **Branch coverage:** % of decision points in the code executed by the tests
- **Function coverage:** % of functions in the code the tests execute
- **Line coverage:** % of lines of code executed by the tests

JAVA Code Coverage tools:



- **JaCoCo** stands for **Java Code Coverage**
- It is a free code coverage library for Java
- It creates code coverage reports and integrates well with CI/CD tools like Azure DevOps, Jenkins, Circle CI, etc.





# Static Code Analysis:

- Static code analysis is the process of examining source code to identify potential defects, security vulnerabilities, and other quality issues
- Static analysis can help you improve the quality and reliability of software by detecting issues early in the development cycle

## Some known Code Analysis Tools:



# SonarQube:

- SonarQube is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code
- It is a self-managed, automatic code review tool that systematically helps you deliver Clean Code

## SonarQube Component:

### SonarLint

Provides immediate feedback in your IDE as you write code so you can find and fix issues before a commit

### PR Analysis

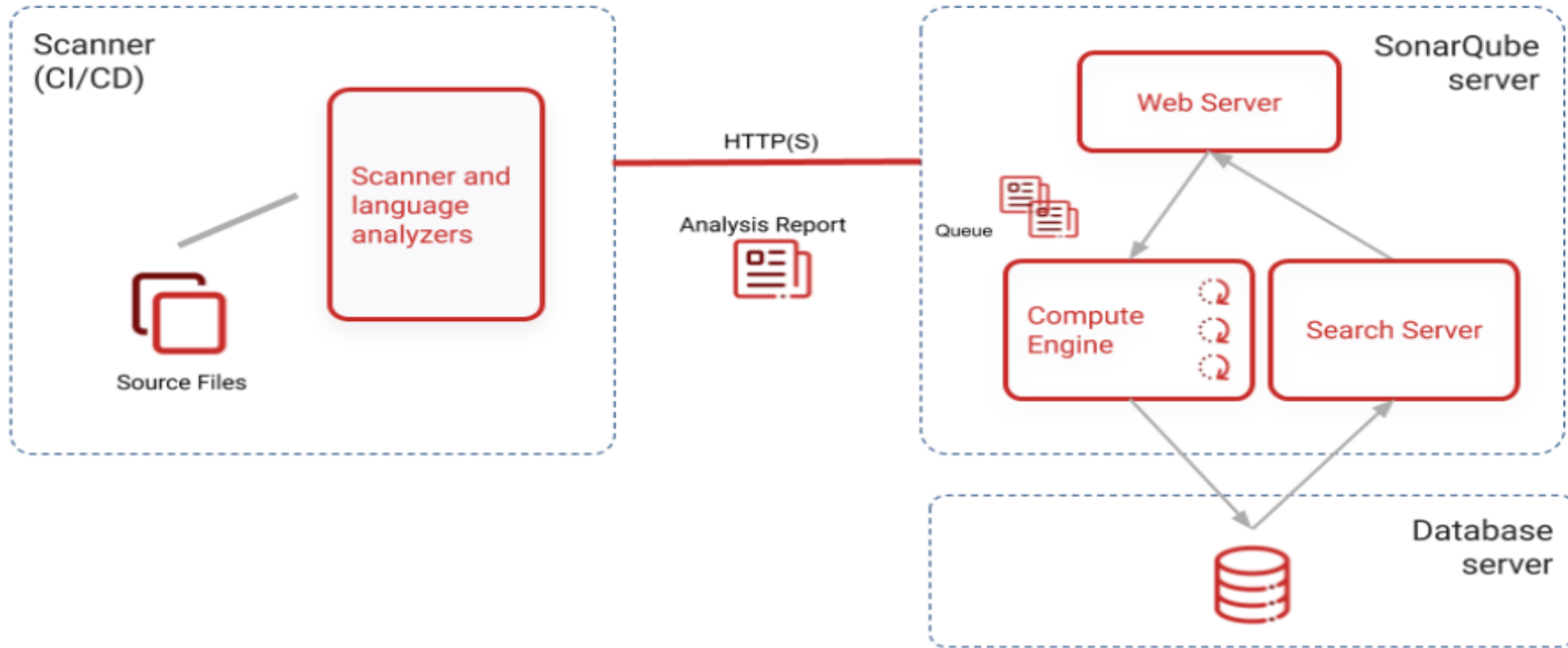
SonarQube's PR analysis fits into your CI/CD workflows with Sonarqube's PR analysis and use of quality gates

### Quality Gates

Keep code with issues from being released to production, a key tool in helping you incorporate the Clean as You Code methodology



# SonarQube Architect:



## **1. The SonarQube server runs the following processes:**

- A web server that serves the SonarQube user interface
- A search server based on Elasticsearch (log analytics, full-text search, security intelligence, business analytics, and operational intelligence use cases)
- The compute engine in charge of processing code analysis reports and saving them in the SonarQube database

## **2. The database to store the following:**

- Metrics and issues for code quality and security generated during code scans
- The SonarQube instance configuration

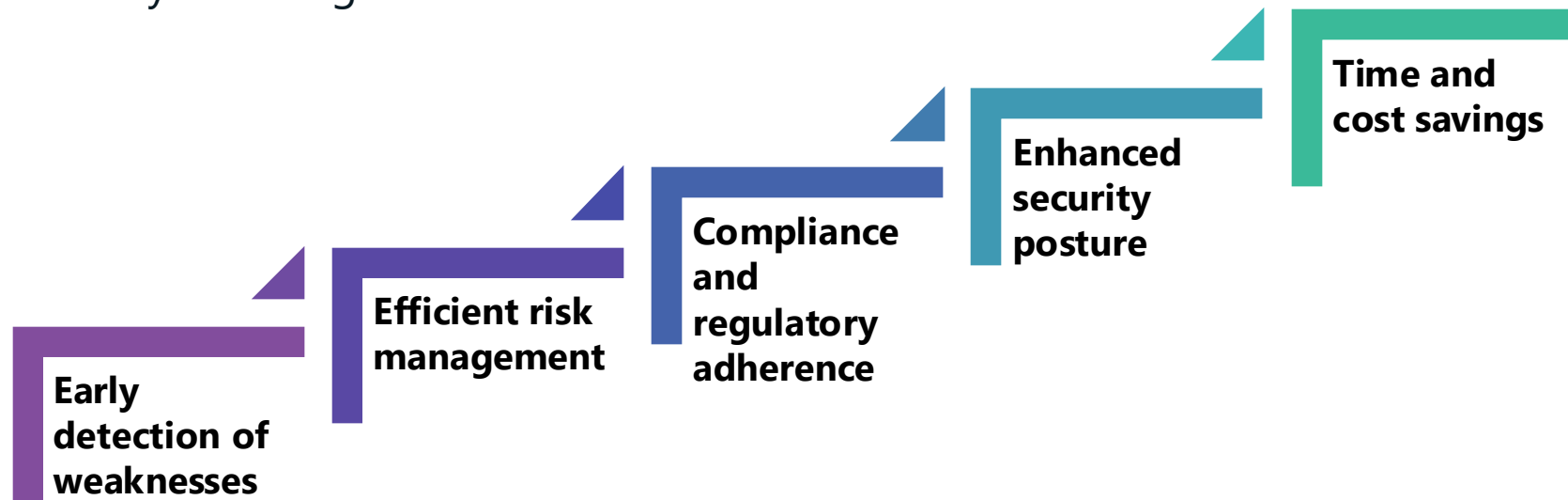
3. One or more scanners running on your build or continuous integration servers to analyze projects



# Vulnerability Scan:

- Vulnerability scanning is the process of discovering, analyzing, and reporting on security flaws and vulnerabilities
- Tools can identify potential risk exposures and attack vectors across an organization's networks, hardware, software, and systems

Benefits of vulnerability scanning:



# Mend Bolt (formerly White Source)

- Mend Bolt is an open-source vulnerability scanning tool within Azure DevOps, which provides:

Real-time  
security alerts

Compliance  
Issues

Licenses  
Compliance

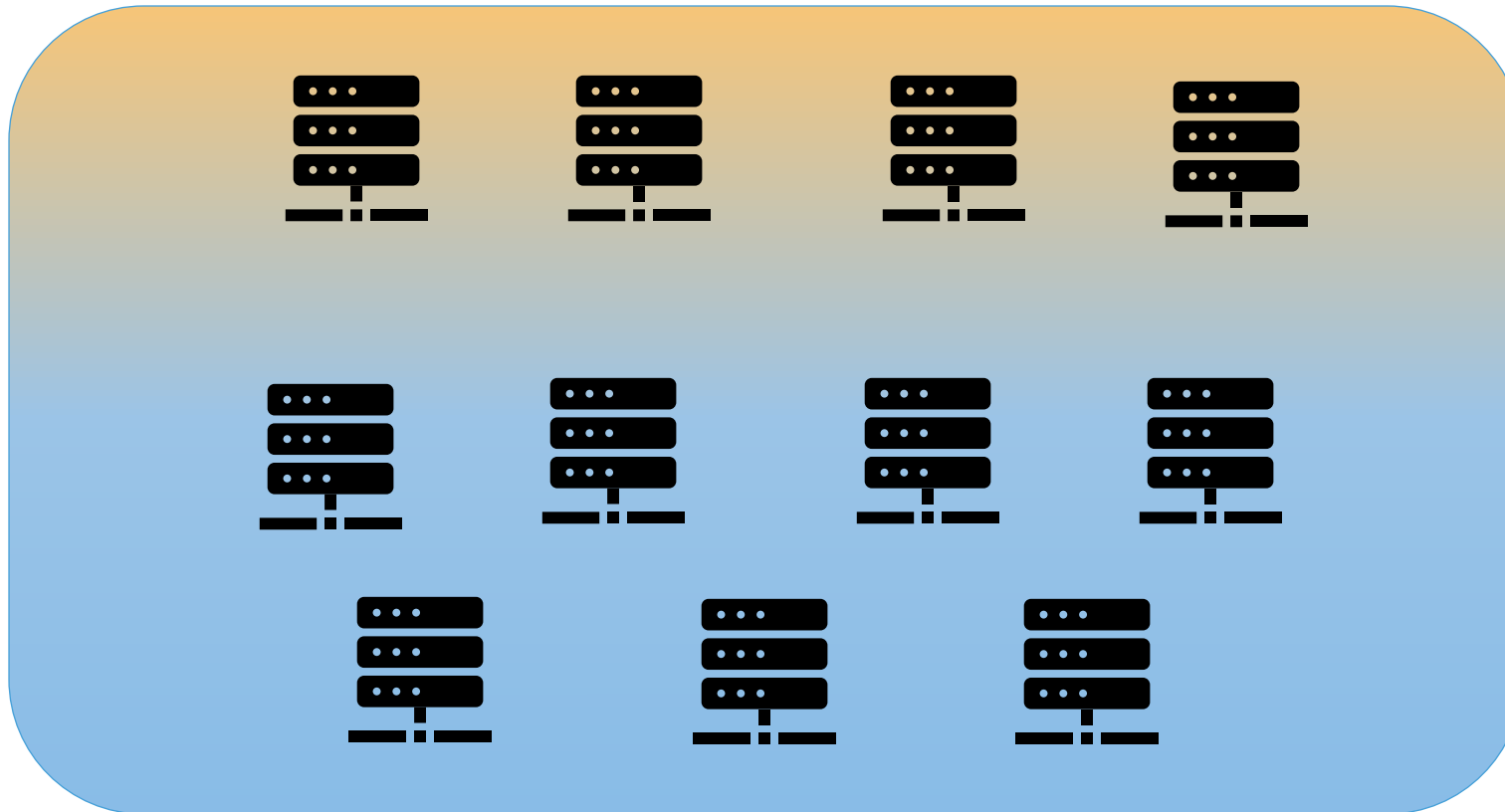
Known  
vulnerabilities



- Deployment Strategies:**
- A deployment strategy is a way to change or upgrade an application
  - The aim is to make the change without downtime in a way that the user barely notices the improvements

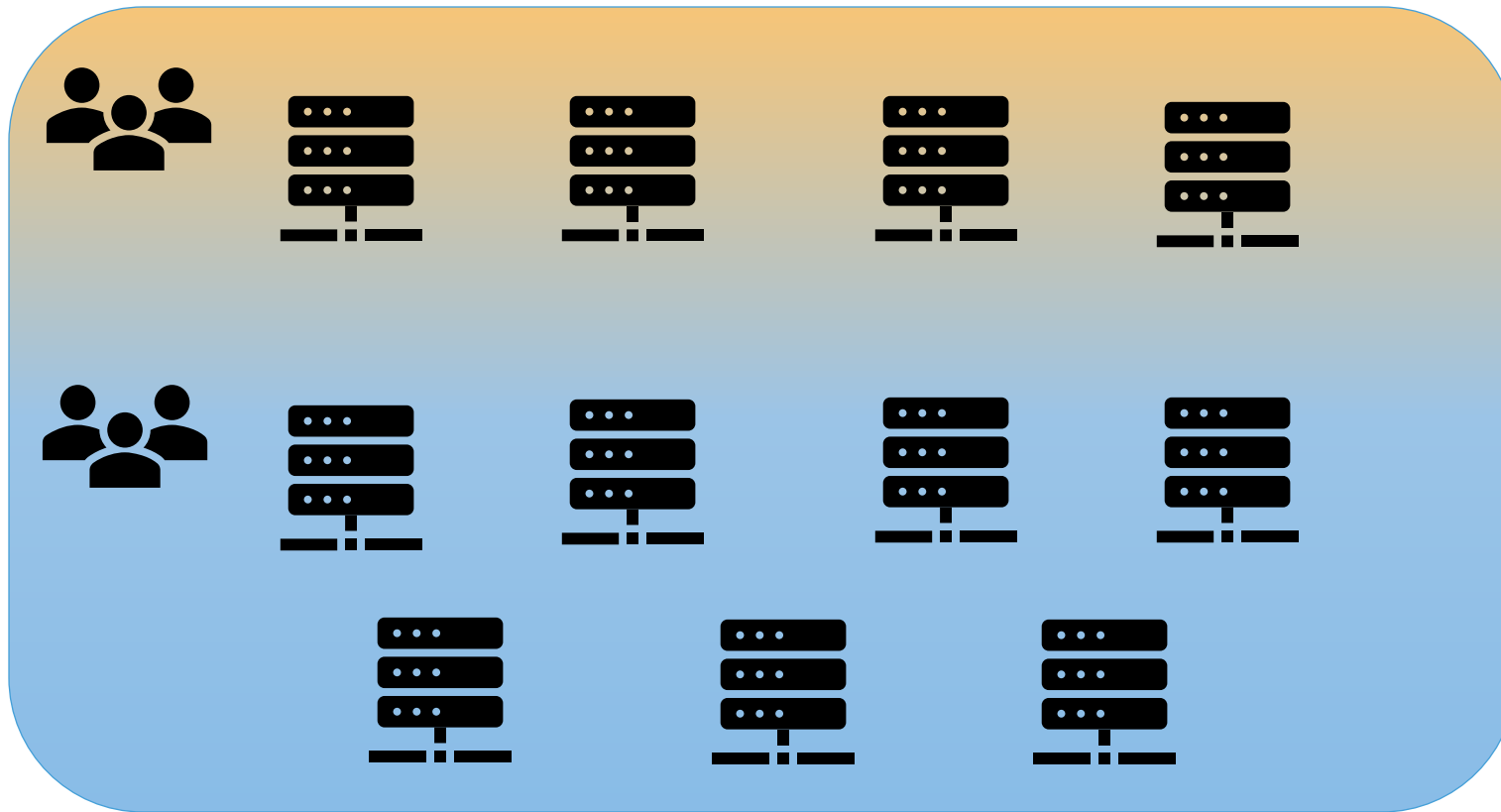
**runOnce** is the simplest deployment strategy wherein all the lifecycle hooks are executed once

**Rolling** deployment replaces instances of the previous version of an application with instances of the new version of the application on a fixed set of virtual machines (rolling set) in each iteration.



# Types of Strategies:

**Canary** by using this strategy, you can roll out the changes to a small subset of servers or users first. As you gain more confidence in the new version, you can release it to more servers in your infrastructure and route more traffic to it



# Deployment - Lifecycle Hooks:

- When you're deploying application updates, the technique you use to deliver the update must :
  - Enable initialization
  - Deploy the update
  - Route traffic to the updated version
  - Test the updated version after routing traffic
  - In case of failure, run steps to restore to the last known good version

The above can be achieved by using lifecycle hooks that can run **steps** during deployment:

preDeploy: Used to run steps that initialize resources before application deployment starts

deploy: Used to run steps that deploy your application

routeTraffic: Used to run steps that serve the traffic to the updated version

postRouteTraffic: Used to run the steps after the traffic is routed. Typically, these tasks monitor the health of the updated version for defined interval

on: failure or on: success: Used to run steps for rollback actions or clean-up

## **Steps:**

A deployment job is a collection of steps that are run sequentially against the environment in YAML Pipelines





# Environments:

- Azure DevOps environments are defined as “a collection of resources that you can target with deployments from a pipeline”
- While an environment is a grouping of resources, the resources themselves represent actual deployment targets
- The Kubernetes resource and virtual machine resource types are currently supported

## **Benefits:**



DEMO



# SQL Database Automation

By: Rohit K Singh



# Agenda:

- Introduction to DAPAC and BACPAC
  - DML & DDL
- Pipeline creation for database automation – Demo
- Azure Vault



# DACPAC vs BACPAC

DACPAC is a Data-Tier Application Package	BACPAC is Backup Package
DACPAC has only the schema and not the data	A BACPAC includes the schema and data from the database
The primary use of DACPAC is to move the tested schema from the test environment to the production environment or bringing the production schema back to the test environment	It is mainly Imports from BACPAC to the database and Exports from the database to BACPAC
DACPAC operations include EXTRACT, DEPLOY, REGISTER, UNREGISTER, UPGRADE  --  DACPAC supports only DDL (like create, drop, rename, and alter) operations and not DML operations (Insert, update, delete)	BACPAC operations include EXPORT, IMPORT
Extension for DACPAC file is .dacpac	Extension for BACPAC file is .bacpac



# DEMO flow:

## Initial Setup - Manual

- Create Azure SQL Server and Database on the cloud
- Enable firewall rules to access the server locally
- Manual deploy scripts using Azure Query editor
- Create initial DACPAC using the SSMS tool locally

## Pipeline Setup for Automation

- Create Azure Repos for database project
- Create a Visual Studio database project and import previously created DACPAC
- Create CI/CD pipelines to automate database deployment

## Database Automation - Test

- Add a new table script in the DACPAC project
- Create a Script folder and add .sql scripts
- Commit and Push changes to the remote repo
- Auto CI/CD pipeline will deploy changes on the Cloud SQL Server



# Azure Vault:

- Azure Key Vault is a cloud service for securely storing and accessing secrets
- It helps us to solve the following problems:

## Secrets Management

Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets

## Key Management

Azure Key Vault can be used as a Key Management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data

## Certificate Management

Azure Key Vault lets you easily provision, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and your internal connected resources



# Agenda:

- Security Groups – Project/Organization level
- Service Hooks - Demo
- User access level
- Azure DevOps – Pricing





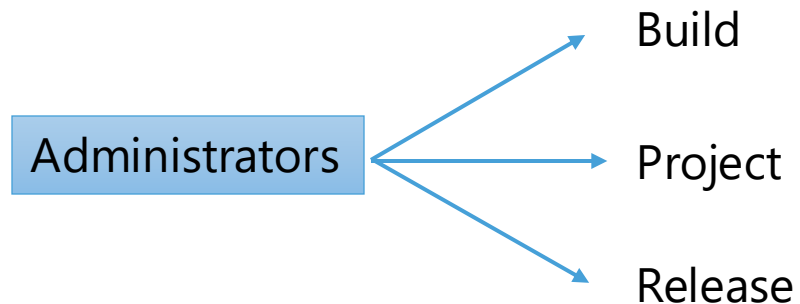
# Security Groups:

## Organization Level:



- ☐ Project Collection Proxy Service Accounts
- ☐ Project Collection Service Accounts
- ☐ Project Collection Test Service Accounts
- ☐ Project Collection Valid Users
- ☐ Project-Scoped Users
- ☐ Security Service Group

## Project Level:

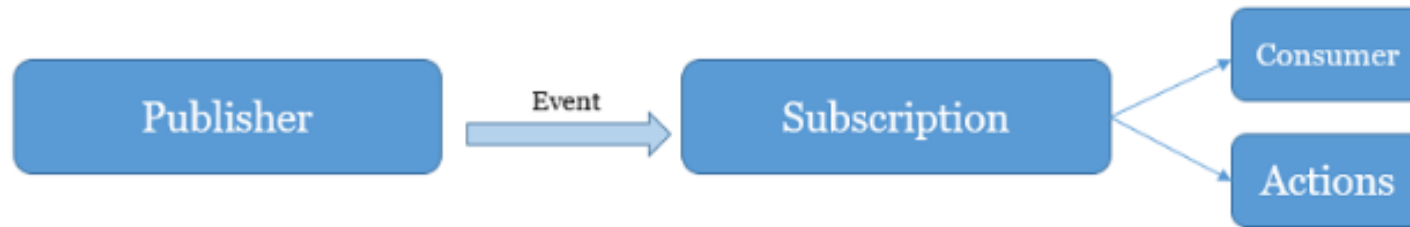


- ☐ Contributors
- ☐ Project Valid Users
- ☐ Readers
- ☐ *TeamName* Team



# Service Hooks:

- Service hooks let you run tasks on other services when events happen in your project in Azure DevOps



- Service hook **publishers** define a set of *events* that you can subscribe to
- **Subscriptions** listen for these *events* and define **actions** to take based on the event
- Subscriptions also target **consumers**, which are external services that can run their actions when events occur



# User Access Level:

- All users added to Azure DevOps are assigned to an *access level*, which grants or restricts access to select web portal features

Three main access levels:

## Stakeholder

- **Stakeholder** access users gain limited access to Azure Boards, Azure Pipelines, and collaboration tools
- They have no access to code repositories

## Basic

- Provides access to most features. Assign to users with a Visual Studio Professional subscription
- An Azure DevOps Server CAL, and to users for whom you're paying for Basic access in an organization

## Basic + Test Plans

- Provides access to all features included in **Basic** and Azure Test Plans

## Visual Studio Subscriber

- Assign to users who already have a Visual Studio subscription
  - Visual Studio Enterprise
  - Visual Studio Professional
  - Visual Studio Test Professional



# Azure DevOps - Pricing

## INDIVIDUAL SERVICES

### Azure Pipelines



1 Free Microsoft-hosted CI/CD  
1 Free Self-Hosted CI/CD

Start free

- 1 Microsoft-hosted job with 1,800 minutes per month for CI/CD and 1 self-hosted job with unlimited minutes per month
- \$40 per extra Microsoft-hosted CI/CD parallel job and \$15 per extra self-hosted CI/CD parallel job with unlimited minutes

### Azure Artifacts



2 GiB free,  
then starting at \$2 per GiB

Start free

- Industry-leading NuGet Server
- Support for Maven, npm, and Python packages
- Upstream sources to help protect open-source dependencies
- Integrated with Azure Pipelines
- Sophisticated access controls

## USER LICENSES

### Basic Plan



First 5 users free,  
then \$6 per user per month

Start free

- **Azure Pipelines:** Includes the free offer from INDIVIDUAL SERVICES
- **Azure Boards:** Work item tracking and Kanban boards
- **Azure Repos:** Unlimited private Git repos
- **Azure Artifacts:** 2 GiB free per organization

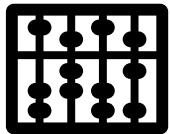
### Basic + Test Plans



\$52 per user  
per month

30 day free trial

- Includes all Basic plan features
- Test planning, tracking & execution
- Browser-based tests with annotation
- Rich-client test execution
- User acceptance testing
- Centralized reporting



# Containerization

By: Rohit K Singh



# Agenda:

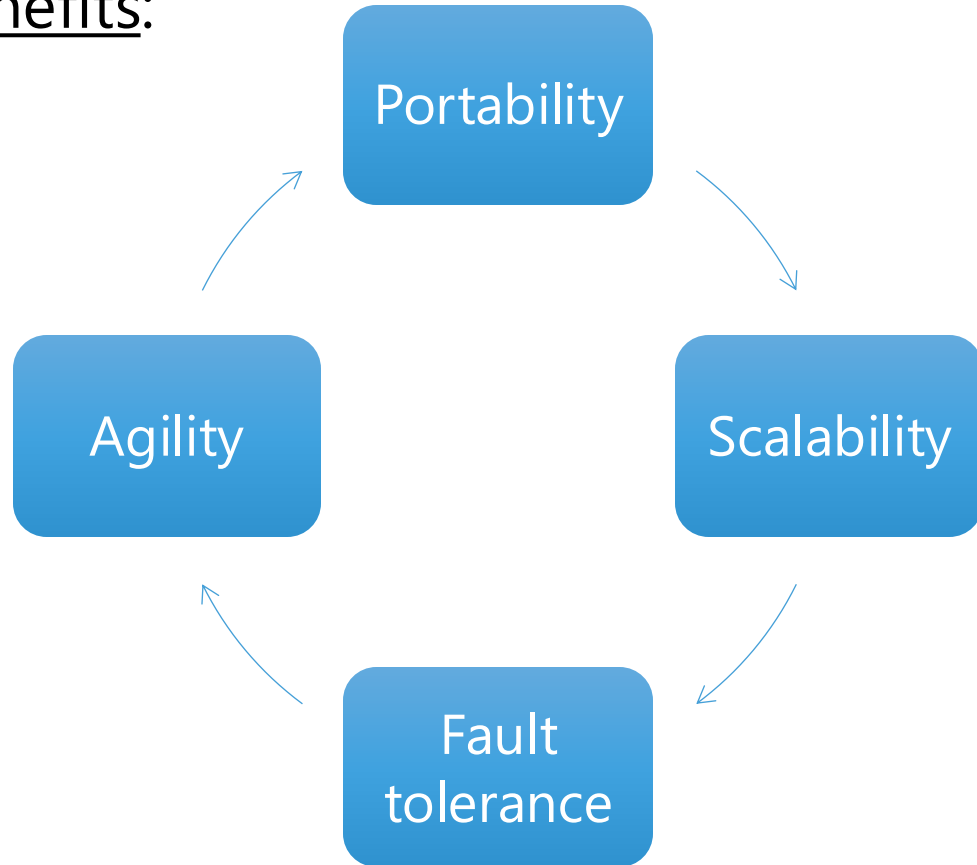
- Containerization – Introduction
- Virtual Machine vs Containers
- Docker – Introduction
- Docker basic commands
- Docker File
- Docker Registry
- Docker Compose
- Demo



# Containerization:

- A container is a lightweight package that bundles together a single app and its dependencies to make applications easier to develop, deploy, and manage across complex IT environments

## Benefits:



## Container Runtime engines:

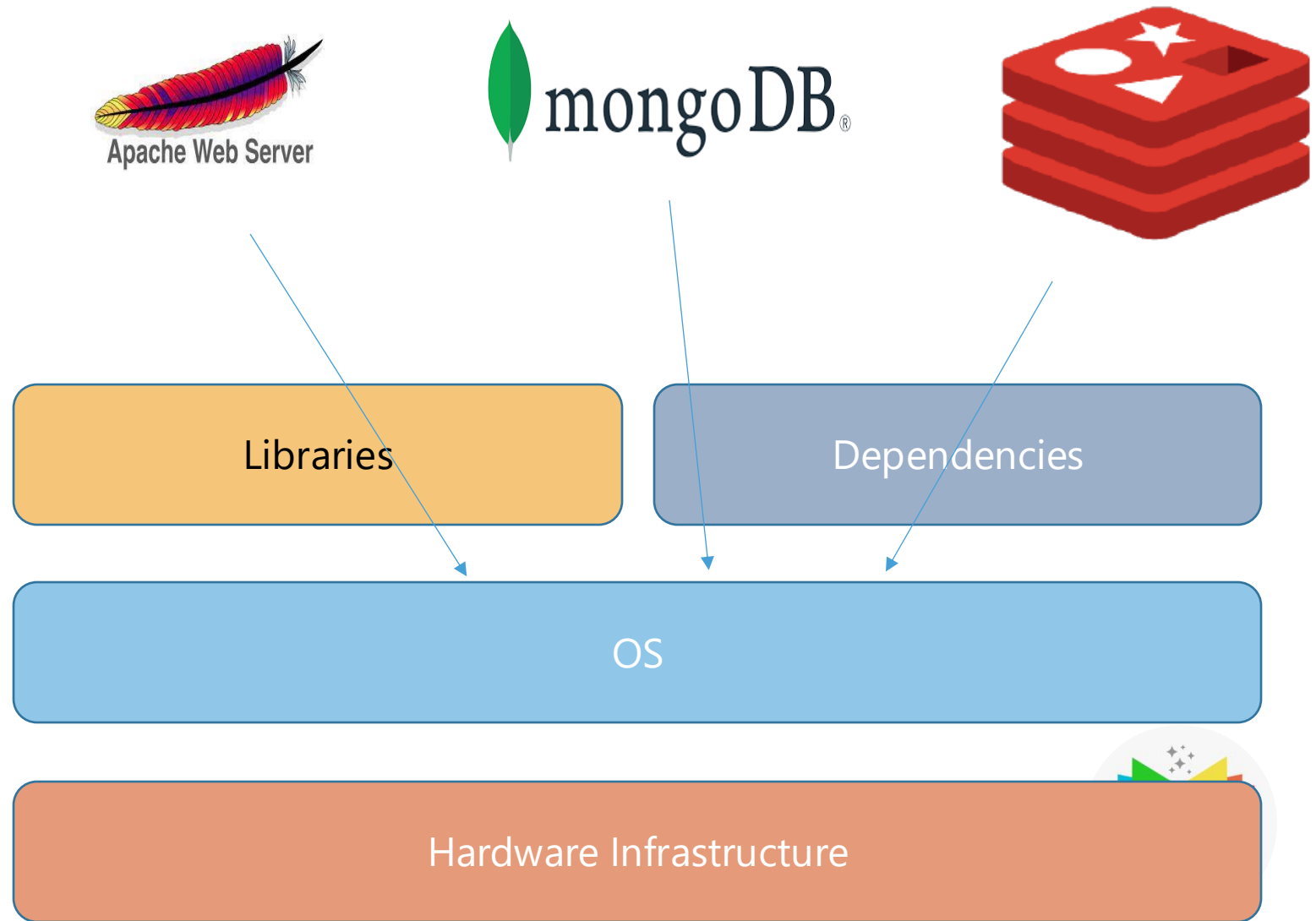
- Docker
- Containerd
- Cri-O
- Rklet
- Kubernetes CRI



# Why do we need Containers?

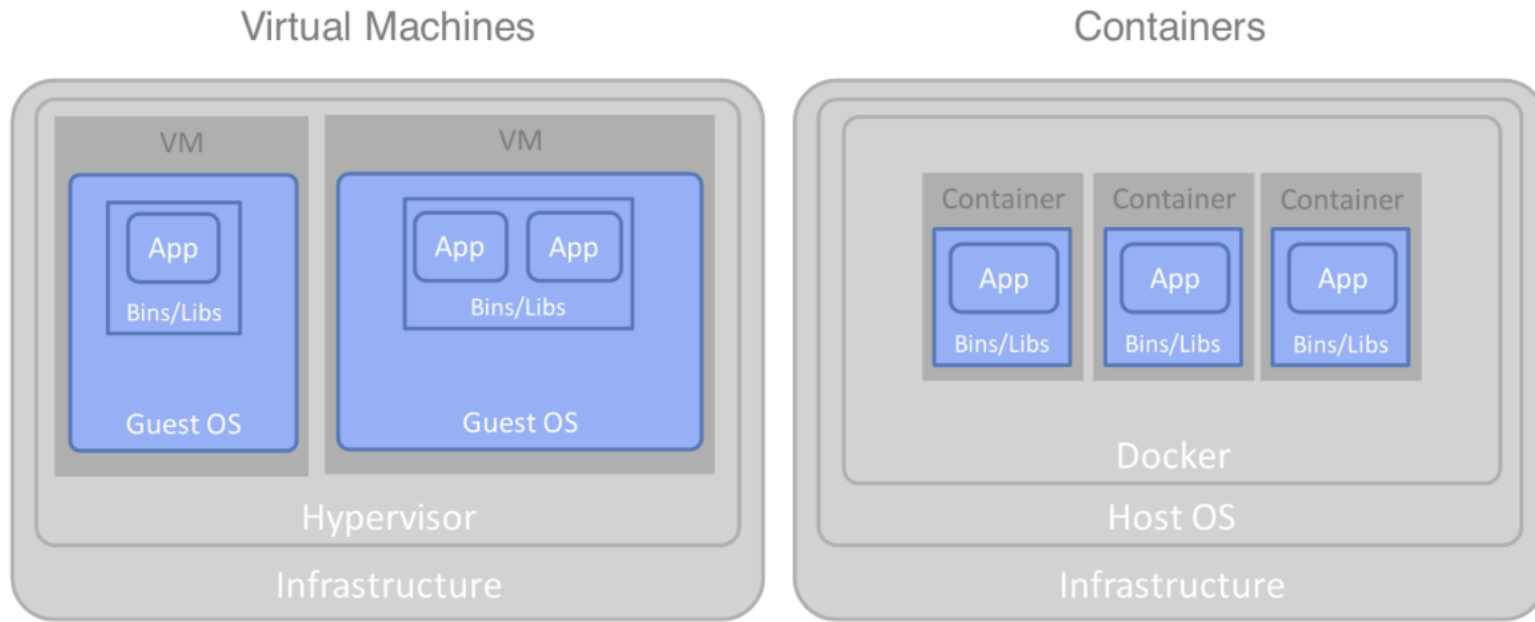
## Problem Statement:

- Compatibility/Dependency
- Long setup time
- Environment configuration drift





# Virtual Machine vs Containers:



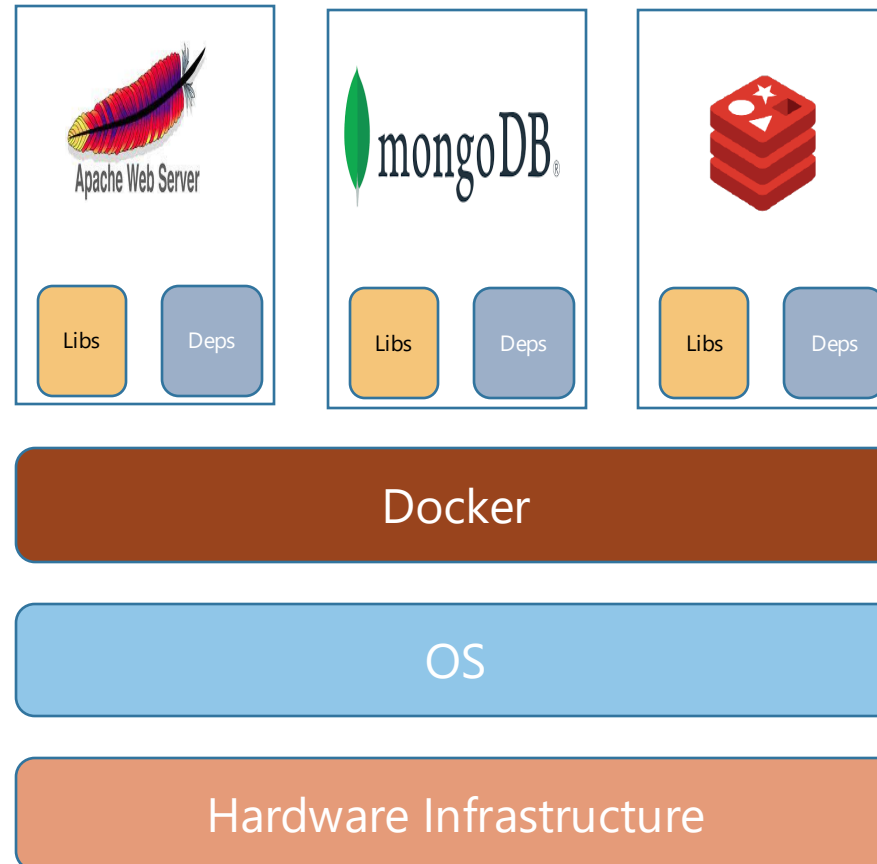
- Container images are lightweight as compared to a VM image
- Eventually taking up less space on disk and memory
- Container boot-ups and down much more rapidly



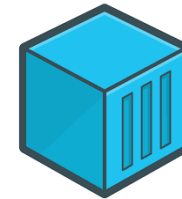
# Docker:

- Docker is a software platform that allows you to build, test, and deploy applications quickly
- Docker packages software into standardized units called containers that have everything the software needs to run including libraries, system tools, code, and runtime

- Containerize Applications
- Run each service with its own dependencies in separate containers



- Processes
- Network
- Mounts



# Docker File:

- A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image
- Docker can build images automatically by reading the instructions from a Dockerfile

The following INSTRUCTION Dockerfile supports:

Format of a Dockerfile:

# Comment

INSTRUCTION arguments

<b><u>ADD</u></b>	Add local or remote files and directories.
<b><u>ARG</u></b>	Use build-time variables.
<b><u>CMD</u></b>	Specify default commands.
<b><u>COPY</u></b>	Copy files and directories.
<b><u>ENTRYPOINT</u></b>	Specify default executable.
<b><u>ENV</u></b>	Set environment variables.
<b><u>EXPOSE</u></b>	Describe which ports your application is listening on.
<b><u>FROM</u></b>	Create a new build stage from a base image.
<b><u>HEALTHCHECK</u></b>	Check a container's health on startup.
<b><u>LABEL</u></b>	Add metadata to an image.
<b><u>MAINTAINER</u></b>	Specify the author of an image.
<b><u>ONBUILD</u></b>	Specify instructions for when the image is used in a build.
<b><u>RUN</u></b>	Execute build commands.
<b><u>SHELL</u></b>	Set the default shell of an image.
<b><u>STOPSIGNAL</u></b>	Specify the system call signal for exiting a container.
<b><u>USER</u></b>	Set user and group ID.
<b><u>VOLUME</u></b>	Create volume mounts.
<b><u>WORKDIR</u></b>	Change working directory.

Sample Dockerfile:

```
FROM mcr.microsoft.com/dotnet/core/sdk:3.1
WORKDIR /app
COPY . .
ENV ASPNETCORE_URLS http://*:5000
EXPOSE 5000
ENTRYPOINT ["dotnet", "azure-devops-webapp.dll"]
```



# Docker Registry:

- A Docker registry is a system for versioning, storing, and distributing Docker images

## ❑ Docker Hub:

- Docker Hub is a container registry built for developers and open-source contributors to find, use, and share their container images
- With Hub, developers can host public repos that can be used for free, or private repos for teams and enterprises

## ❑ Amazon ECR:

- Amazon Elastic Container Registry (Amazon ECR) is an AWS-managed container image registry service that is secure, scalable, and reliable
- Amazon ECR supports private repositories with resource-based permissions using AWS IAM

## ❑ Azure Container Registry ACR:

- Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments
- ACR gets easily integrated with Azure CI/CD pipelines



# Docker Compose:

- Docker Compose is a tool that helps you define and share multi-container applications
- With Compose, we can create a YAML file to define the services and with a single command, we can spin everything up or tear it all down

```
docker compose up -d
```

```
docker compose down
```

## Benefits of Docker Compose:

- **Single host deployment** - This means you can run everything on a single piece of hardware
- **Quick and easy configuration** - Due to YAML scripts
- **High productivity** - Docker Compose reduces the time it takes to perform tasks
- **Security** - All the containers are isolated from each other, reducing the threat landscape



DEMO



# Kubernetes

By: Rohit K Singh



# Agenda:

- Container Management – Introduction
- Kubernetes – Introduction
- Kubernetes – Architecture
- Types of Services
- Volumes
- Deployment Strategies
- Taint & Toleration
- Demo

- Azure Kubernetes Services – Introduction
- Components of AKS Cluster
- Demo



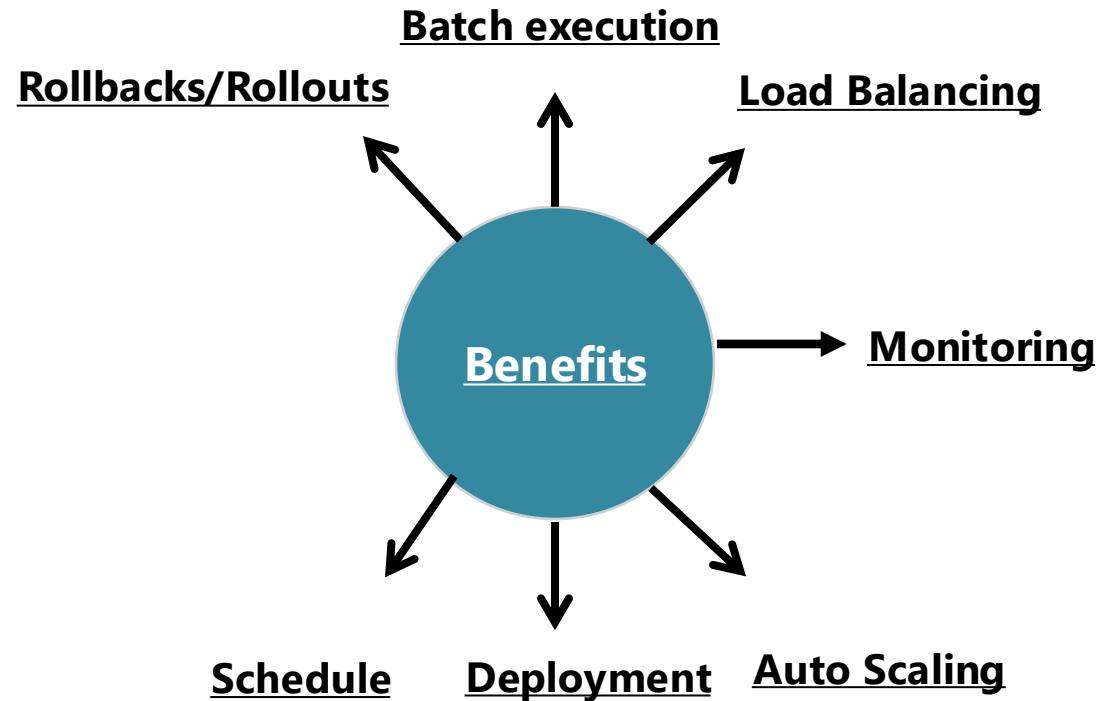


# Container Management:

- Container management is a process for automating the creation, deployment, and scaling of containers
- Container management facilitates the addition, replacement, and organization of containers on a large scale

Container Runtime engines:

- Docker
- Containerd
- Cri-O
- Rklet
- Kubernetes CRI



Few well-known container management tools:

- ☐ Kubernetes    ☐ Docker Swarm
- ☐ Apache Mesos Marathon    ☐ OpenShift



# Kubernetes:

- Container management (orchestration) tool
- Developed by Google Lab (CNCF) Cloud Native Computing Foundation
- Open Source
- Written on Golang
- Also known as K8s

## Automatic bin-packing

- It automatically packages your application and schedules the container based on the requirements and resources available

## Networking

- It can control over network and communication between pods and can load balance across them

## Storage

- AWS (EBS) AWS S3/NFS/Azure (Blob) -- It allows you to mount the storage system of your choice



## Self - healing

- if the container fails -> restart the container
- if a node dies -> replace and reschedule containers for other nodes
- if the container does not respond to the user-defined health check -> kill the container Self-healing

## Automatic Rollbacks and Rollout

- Rollout -> Deploy changes to the application or its configuration
- Rollback -> Revert the change and restore to the previous state
- Zero downtime -> No downtime while rolling back the change

## Secrets

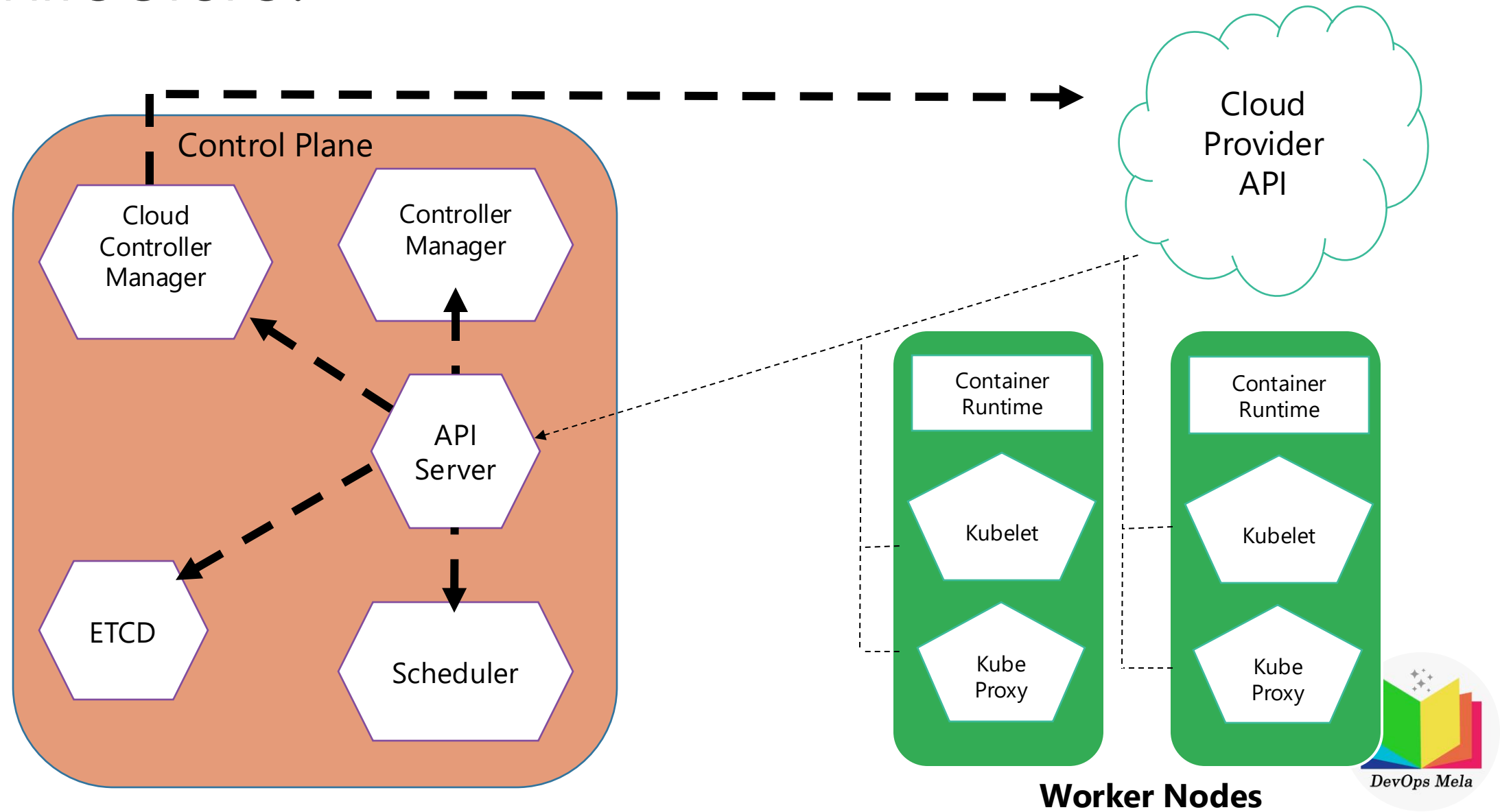
- Sensitive data like passwords, keys, and tokens are handled using secrets
- A secret is an object in k8s
- It is created outside pods and containers

## Config Maps

- The configuration is handled by Config Maps
- Config Map is a k8s object
- Created outside pods and containers



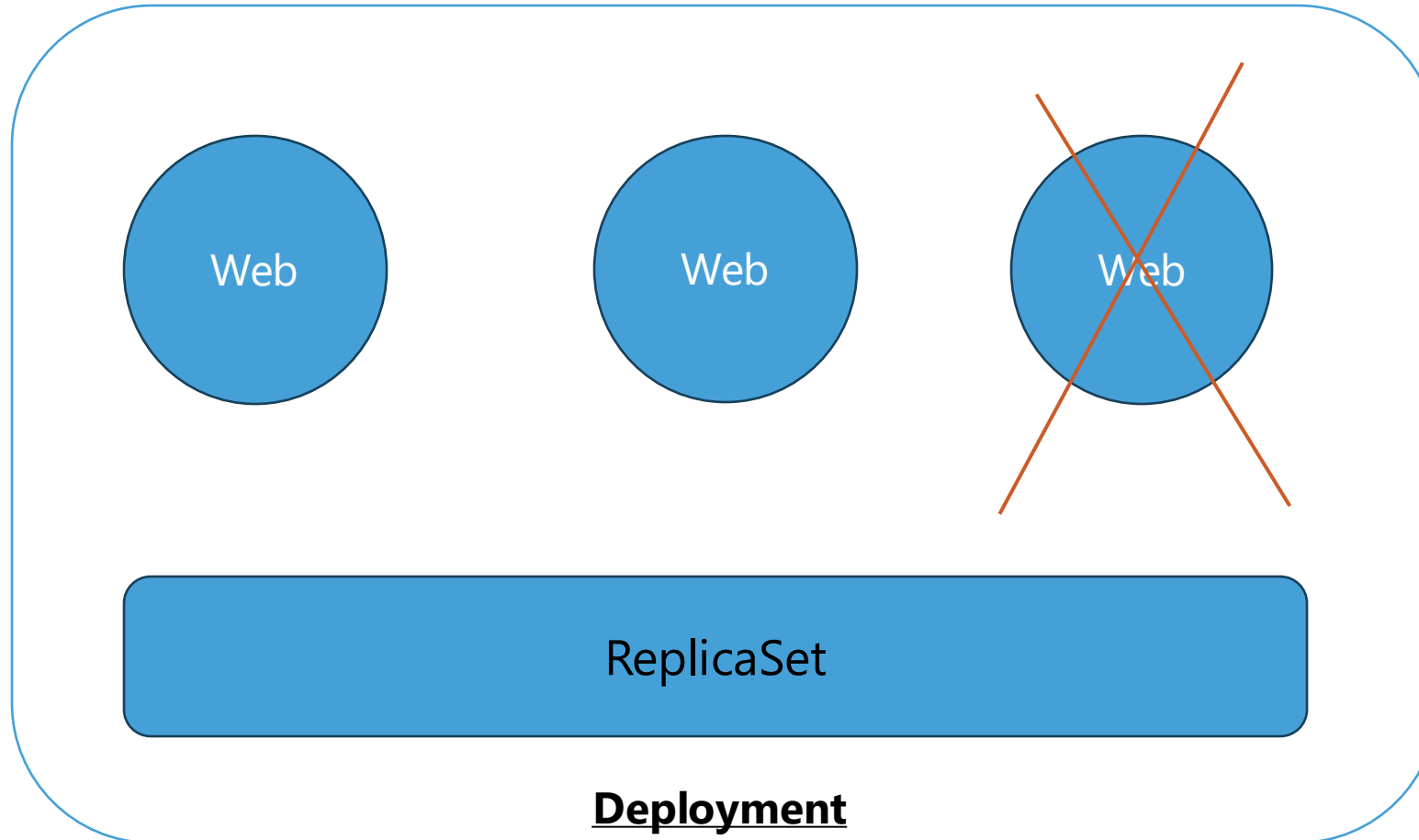
# Architecture:



# Deployment:

- A Kubernetes Deployment tells Kubernetes how to create or modify instances of the pods that hold a containerized application
- Deployments can help to efficiently scale the number of replica pods, enable the rollout of updated code in a controlled manner, or roll back to an earlier deployment version if necessary

## Work Flow:



# Labels & Selectors:

- They are standard method to group things together in Kubernetes
- We can filter the objects based on the criteria like class, kind, and functions
  - Labels are the properties attached to each item/object
  - Selector helps us to filter the items/objects that have labels attached to them

Define labels under metadata of pod-definition:

```
labels:  
  app: app1  
  type: frontend
```

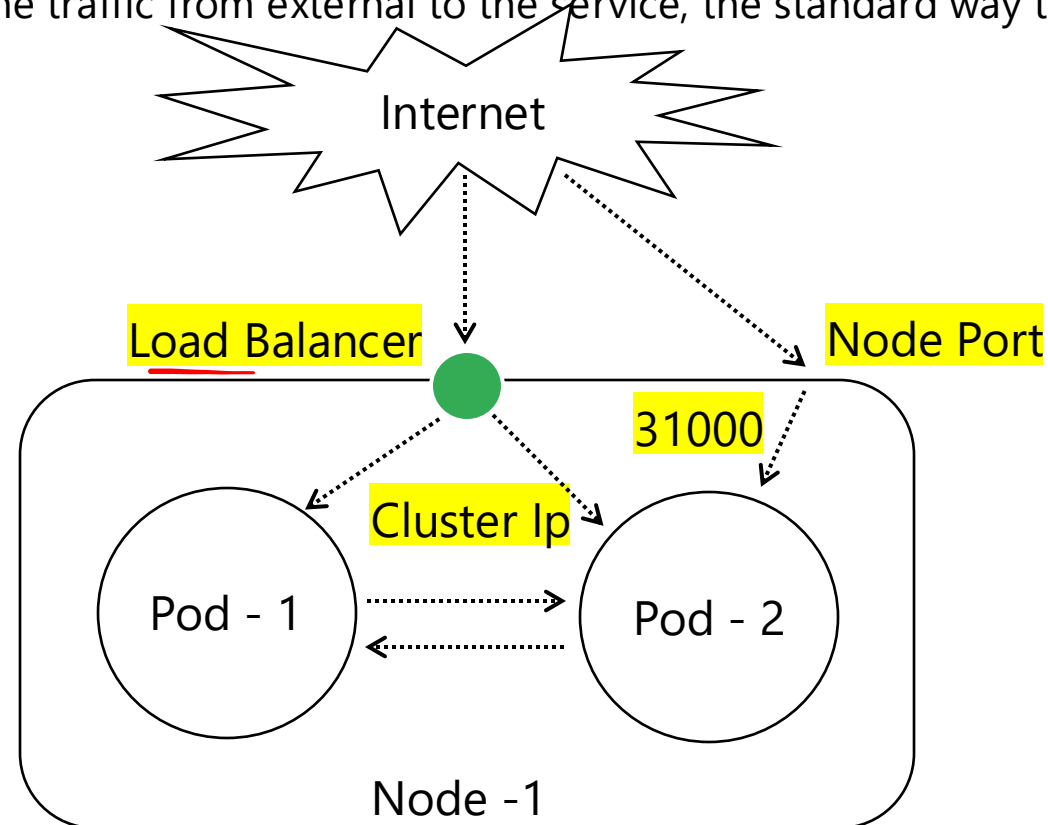
Then while selecting, specify a condition to filter specific objects:

```
selector:  
  matchLabels:  
    app: app1  
    type: frontend
```



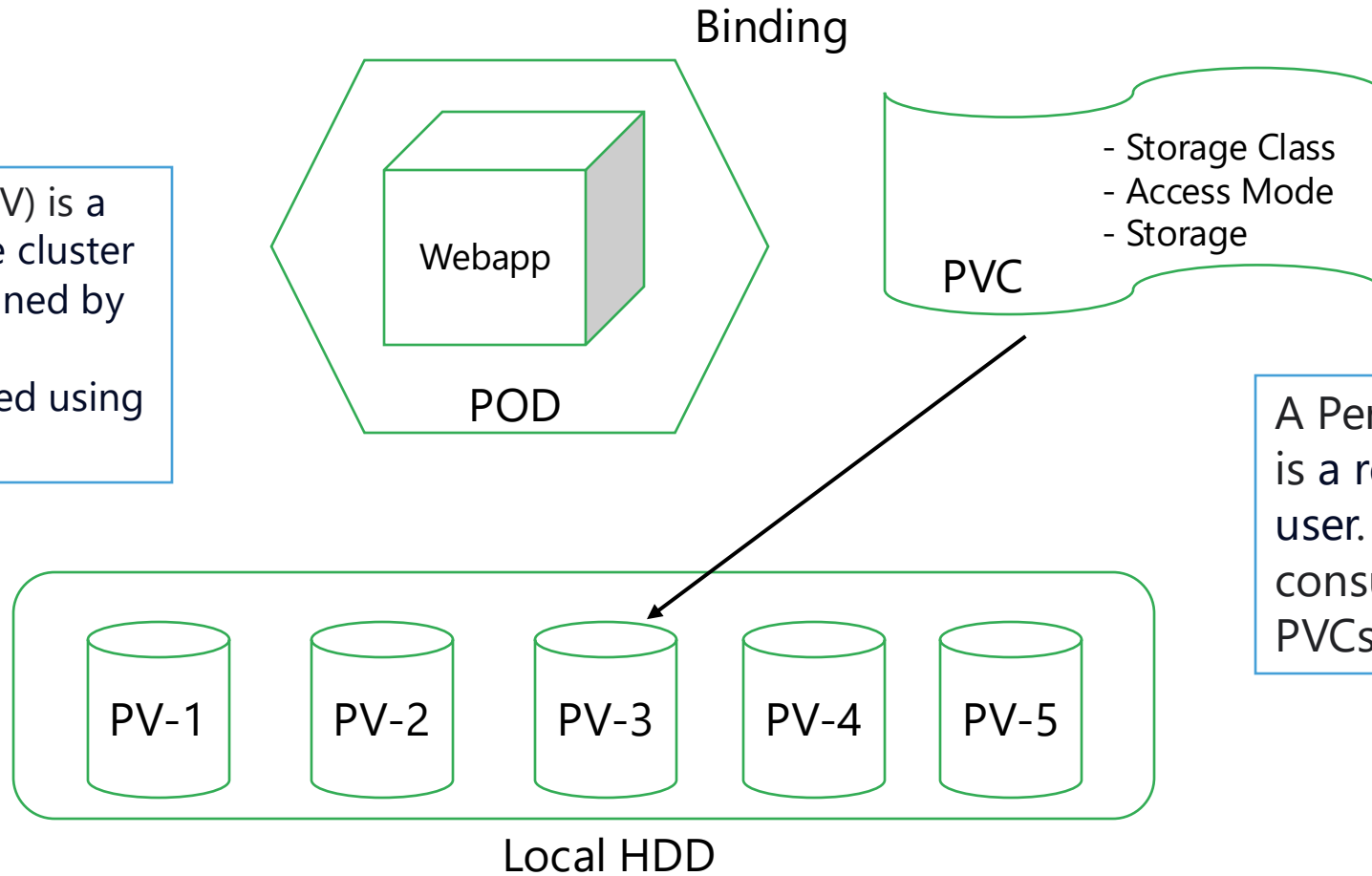
# Services:

- Kubernetes services enable communication between various components within or outside the application
  - **Cluster IP**: Facilitates internal communication with other apps in your cluster, no external access and it's the default Kubernetes service
  - **Node Port**: Open a specific port on the node and forward traffic to the pod via the service, port to choose from is 30000:32767
  - **Load Balancer**: Load Balancer which will route the traffic from external to the service, the standard way to expose your application to the internet



# Volumes:

A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes



A PersistentVolumeClaim (PVC) is a request for storage by a user. It is similar to a Pod. Pods consume node resources and PVCs consume PV resources

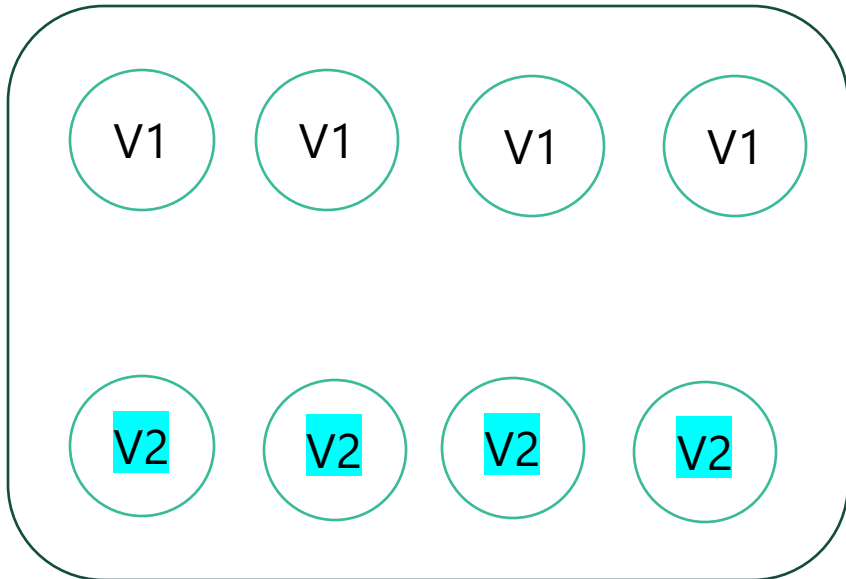




# Deployment Strategy:

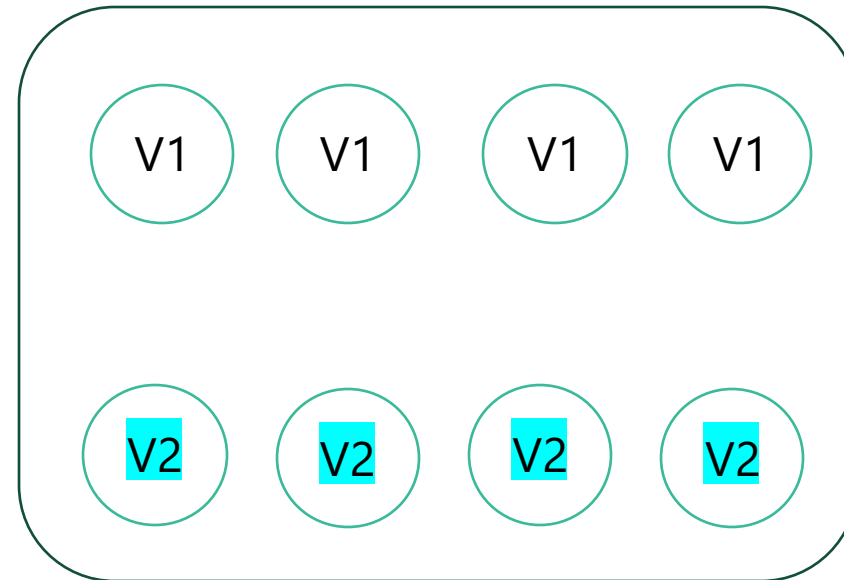
## ➤ Rolling Update Deployment:

- Default deployment strategy in Kubernetes
- It replaces pods, one by one, of the previous version of our application with pods of the new version without any cluster downtime
- A rolling deployment slowly replaces instances of the previous version of an application with instances of the new version of the application



## ➤ Recreate Deployment:

- In recreate deployment, we fully scale down the existing application version before we scale up the new application version



# Taint & Toleration:

- **Taint:**
  - ✓ Taints are the opposite -- they allow a node to repel a set of pods
- **Toleration:**
  - ✓ Tolerations are applied to pods. Tolerations allow the scheduler to schedule pods with matching taints

A taint can produce three possible effects:

➤ **NoSchedule**

The Kubernetes scheduler will only allow scheduling pods that have tolerations for the tainted nodes

➤ **PreferNoSchedule**

The Kubernetes scheduler will try to avoid scheduling pods that don't have tolerations for the tainted nodes

➤ **NoExecute**

Kubernetes will evict the running pods from the nodes if the pods don't have tolerations for the tainted nodes

**Note:**

**Taints** and **Tolerations** work together to ensure that pods are not scheduled onto inappropriate nodes



DEMO



# Azure Kubernetes Services

By: Rohit K Singh



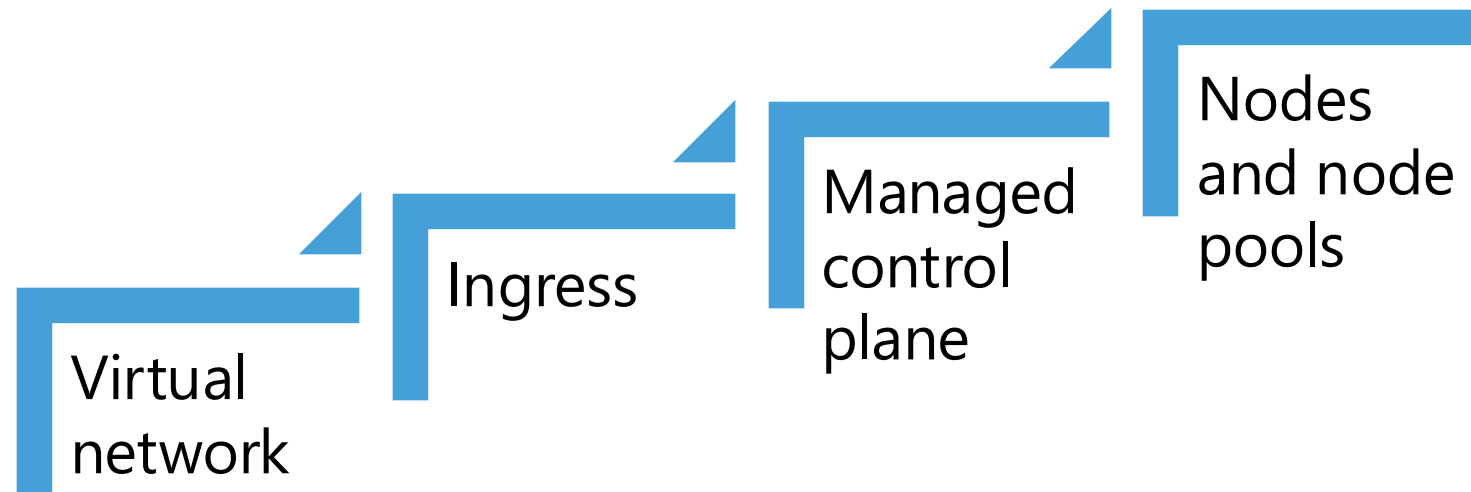
# Azure Kubernetes Services:

- Azure Kubernetes Service is a fully managed, open-source container orchestration service used to deploy, scale, and manage container-based applications on Azure cloud clusters

## Ways to Provisioning AKS Cluster:

- ☐ Azure CLI
- ☐ Azure Portal
- ☐ Azure PowerShell
- ☐ Template-driven options like Terraform or Azure Resource Manager template

## Components of an AKS cluster:



DEMO



# Helm Charts Service Mesh Monitoring

By: Rohit K Singh



# Agenda:

- Helm Charts – Introduction
- Setup and Working of Helm Charts
- Service Mesh - Concept
- Istio – Introduction
- Istio Architecture
- Istio Injection - Demo
- Monitoring Tools
  - Kiali
  - Prometheus
  - Grafana
  - Metric Server
- Demo





# Helm Charts:

- Helm is the package manager for Kubernetes (like yum) that allows easily package, configure, and deploy applications onto Kubernetes clusters

The four main components that are required for a Helm Chart to be executed are as follows:

- ❑ Chart.yaml: It has required information describing what the Chart is about, with a name and version
- ❑ Values.yaml: The Values.yaml file is where you define values to be injected/interpreted by the templates
- ❑ Charts Directory for other Charts: A chart is a collection of files that describe a related set of Kubernetes resources
- ❑ Templates (Directory): The Template Directory is where existing Kubernetes manifests can leverage the values in Values.yaml

Setup & Install:

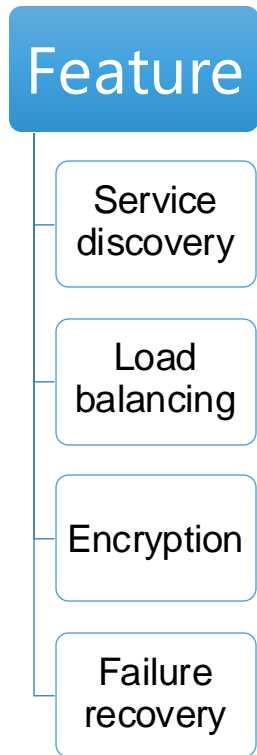
```
$ curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
$ chmod 700 get_helm.sh
$ ./get_helm.sh
```



# Service Mesh

is a dedicated infrastructure layer that controls service-to-service communication over a network

- A service mesh controls the delivery of service requests in an application
- A method enables separate parts of an application to communicate with each other



# Istio

is an open framework for connecting, securing, managing, and monitoring microservices

- It simplifies observability, traffic management, security, and policy with the leading service mesh
- It supports various service-mesh features:

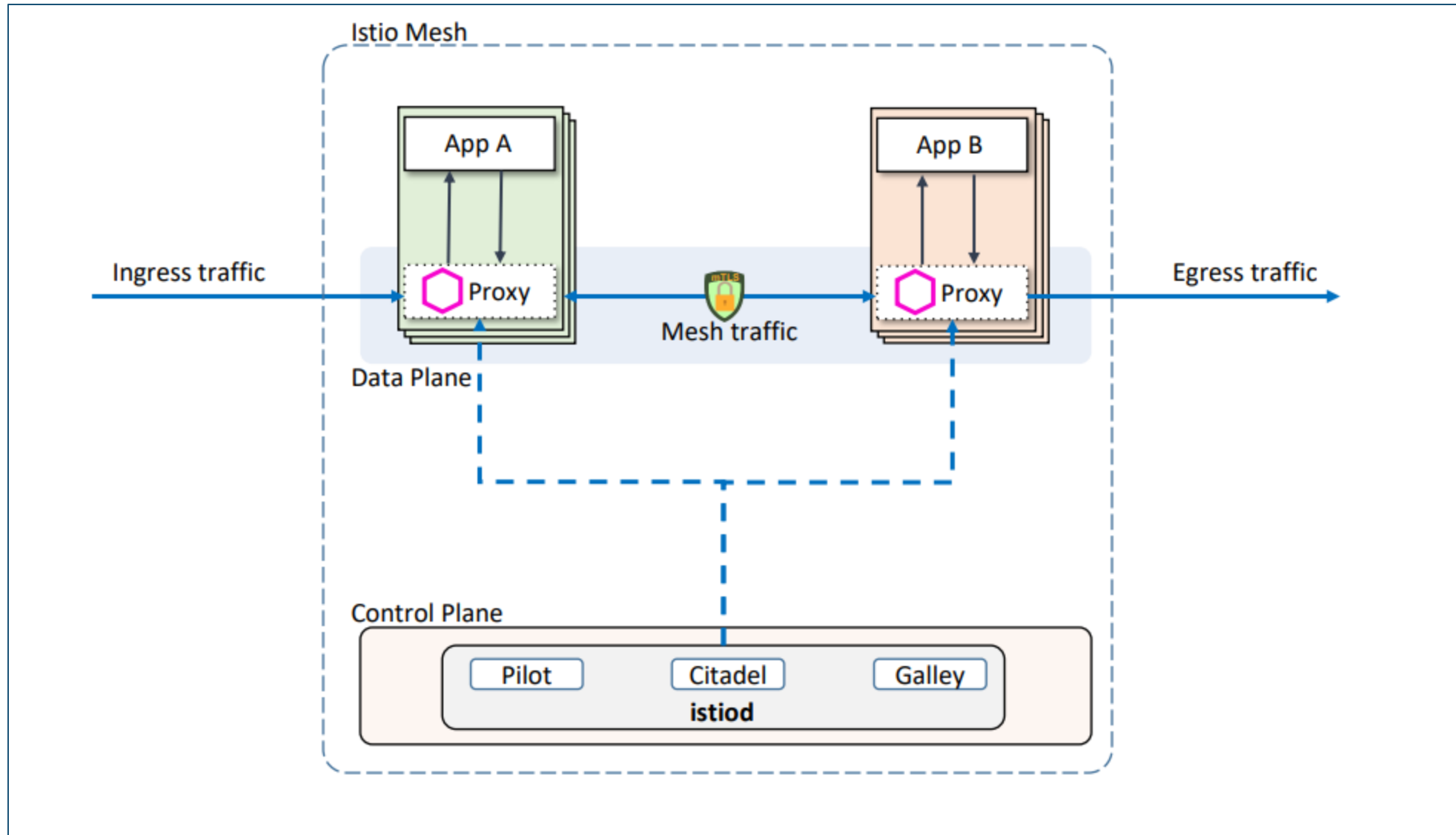
- Discovery
- Load balancing
- Failure recovery
- Metrics
- Monitoring

Istio can also handle more complex operational requirements

- A/B testing
- Canary releases
- Rate limiting
- Dark launches
- Access control
- End-to-end authentication



# Architecture:



# Monitoring:

- Kubernetes has the potential to simplify the process of deploying your application in containers and across clouds, but in doing so, it leaves you blind as to,
  - what is happening,
  - what resources are being utilized

Importantly need to understand what metrics need to be monitored:

- **Node level metrics** – Check if the nodes are healthy
- **Performance level metrics** – like CPU, memory, network, and disk utilization
- **Pod level metrics** – checking healthy pods and performance level metrics of pods {cpu, mem}



- The Kubernetes Metrics Server is a cluster-wide aggregator of resource usage data
- Metrics Server retrieves metrics from each node and pods aggregate them and store them in memory
- As a result, it doesn't store historical data, hence can't be used for production-level cluster



Kubelet → CAdvisor → Performance metrics available for cluster

## Prometheus:

- Prometheus is a free software application used for event monitoring and alerting. It records metrics in a time series database built using an HTTP pull model, with flexible queries and real-time alerting
- It uses PromQL, a powerful query language for querying your time series data
- You can visualize metrics using tools like Grafana and Kiali

## Grafana:

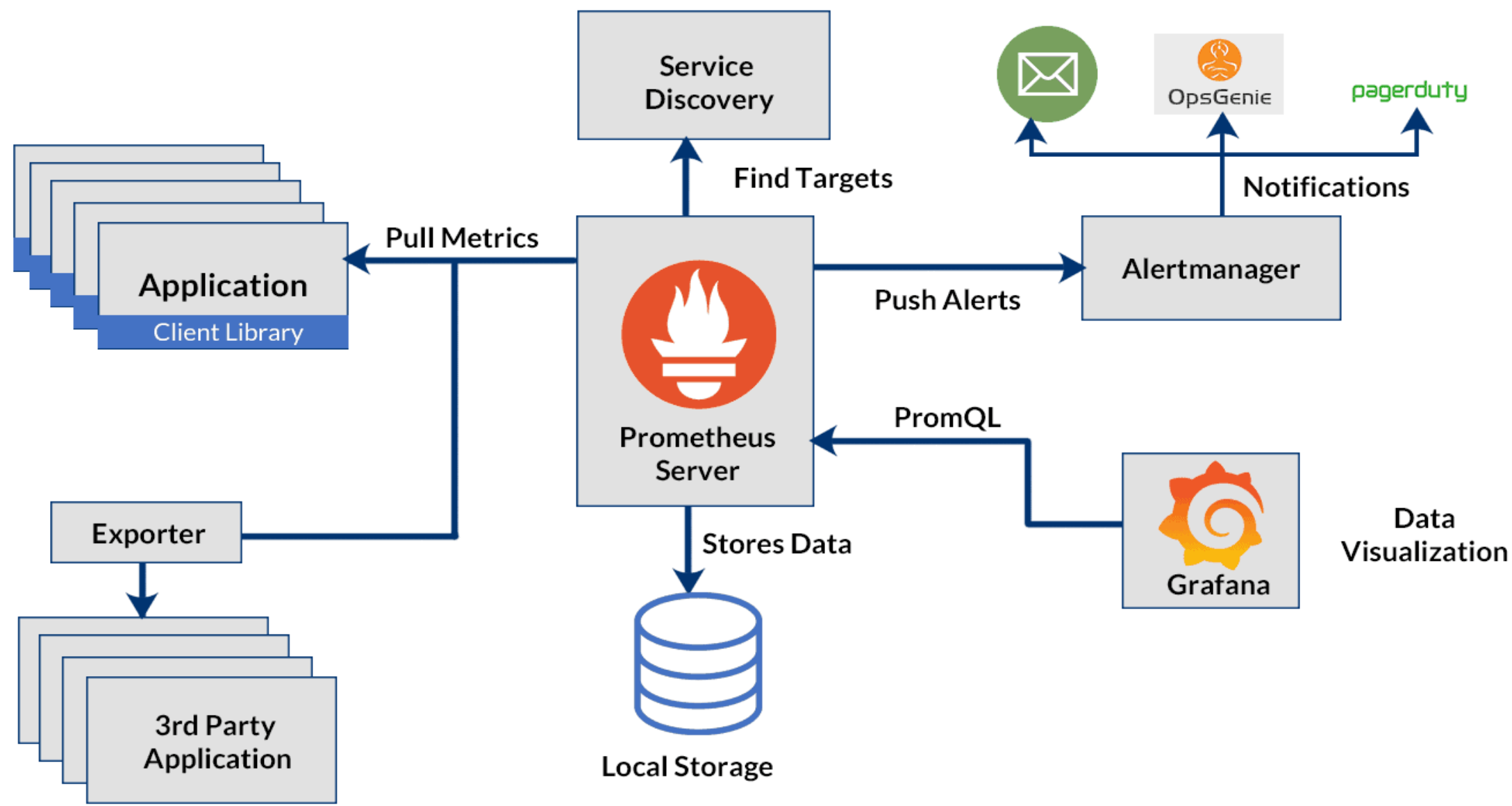
- Grafana open-source software enables you to query, visualize, alert on, and explore your metrics, logs, and traces wherever they are stored
- Grafana provides you with tools to turn your time-series database (TSDB) data into insightful graphs and visualizations
- Grafana plugin framework also enables you to connect other data sources like NoSQL/SQL databases, ticketing tools like Jira or ServiceNow, and CI/CD tooling like GitLab

## Kiali:

- Kiali is an observability console for Istio with service mesh configuration capabilities
- It helps you to understand the structure of your service mesh by inferring the topology and also provides the health of your mesh
- Kiali provides detailed metrics, and a basic Grafana integration is available for advanced queries



# Prometheus Architecture:



DEMO

