Azure DevOps 👩🏻‍💻

Section 1 part 1

Waterfall method

Requirements

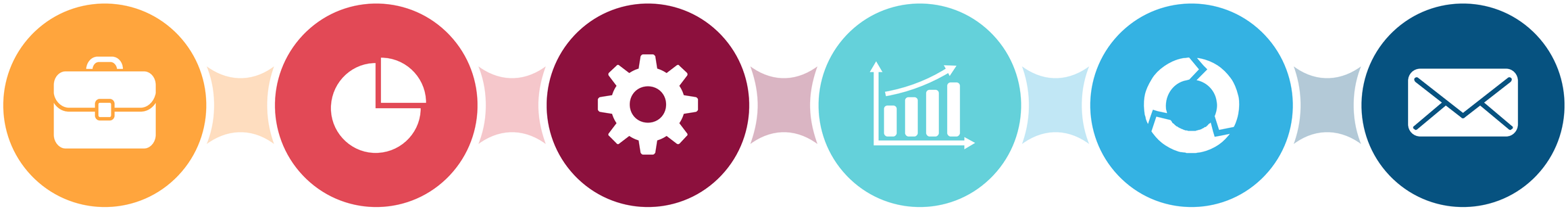
Of the system need be developed are documented

Development

Developers develops the code for the application

Deployment

App is deployed in the customer environment/ Market



Helps in specifying the hardware and system requirements

Design

The entire system is tested for any faults and failures.

Testing

To fix Issues patches are released

Maintenance

Problems by Waterfall Methodology

01

No Software

until late during the life cycle

02

High Risk

Project requirements change

03

Integration

is done as a "big-bang

04

Measure

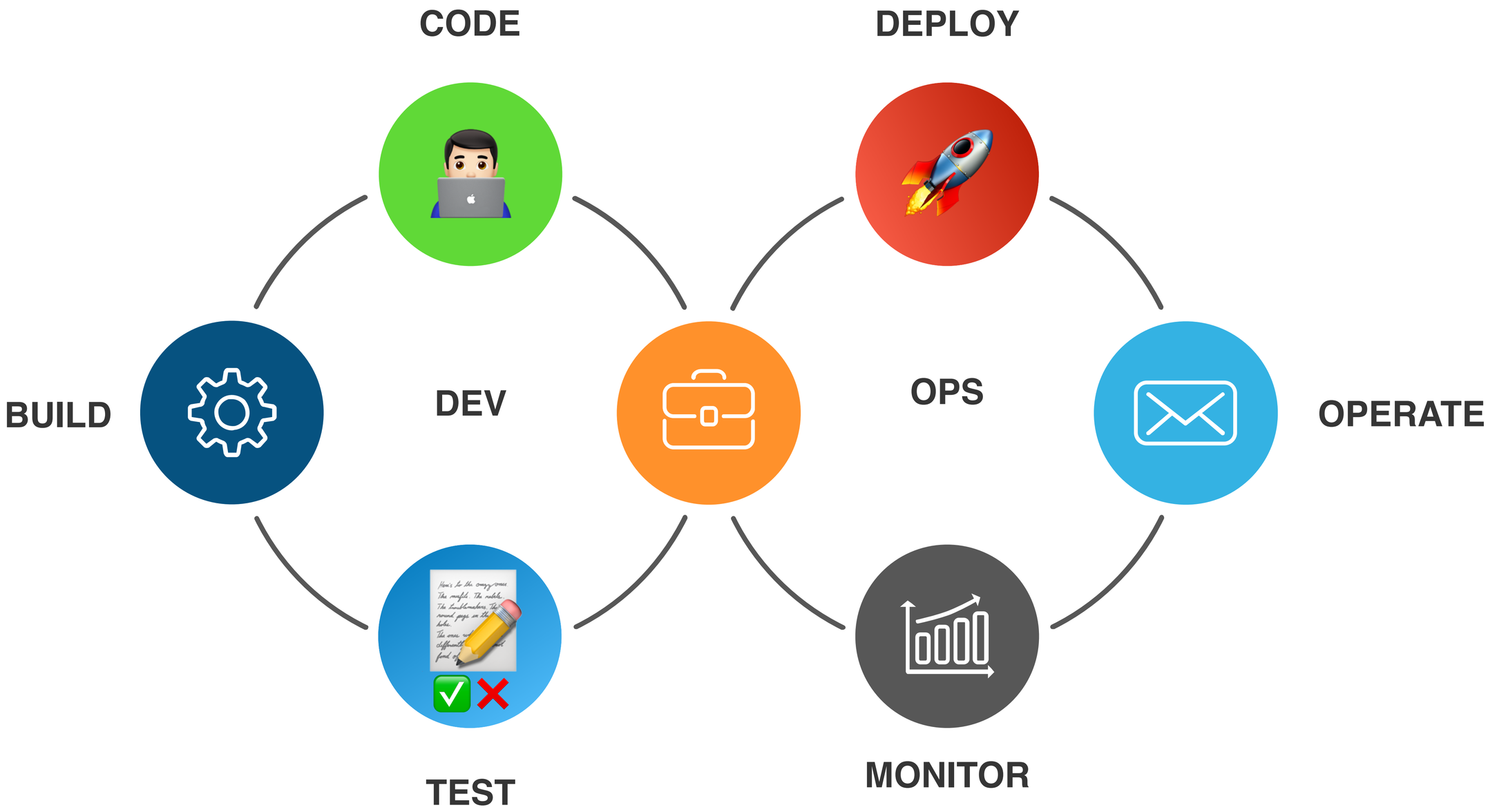
difficult to measure progress within stages.

05

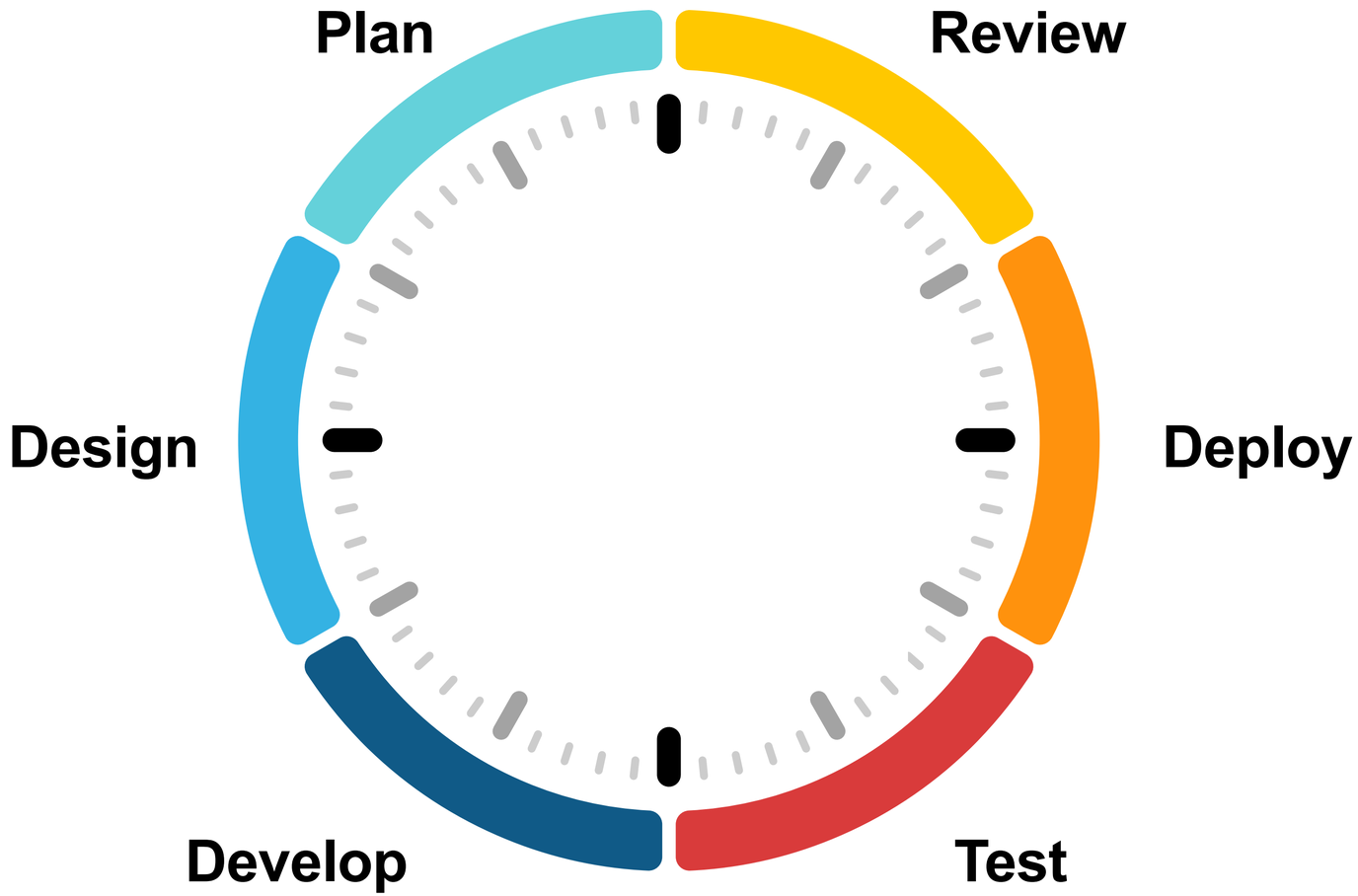
High Costs

In terms of man force and time

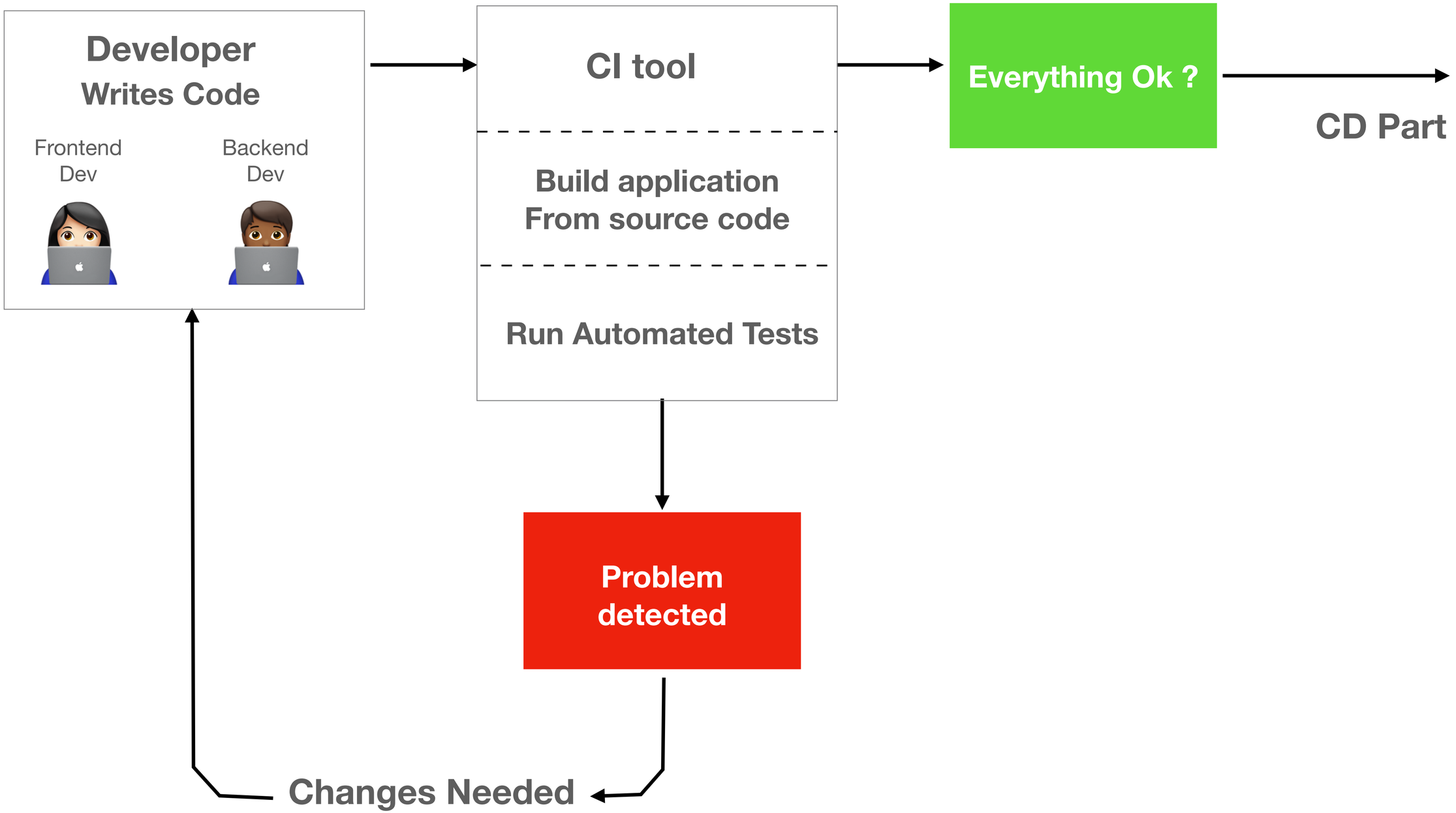
DevOps 🤔

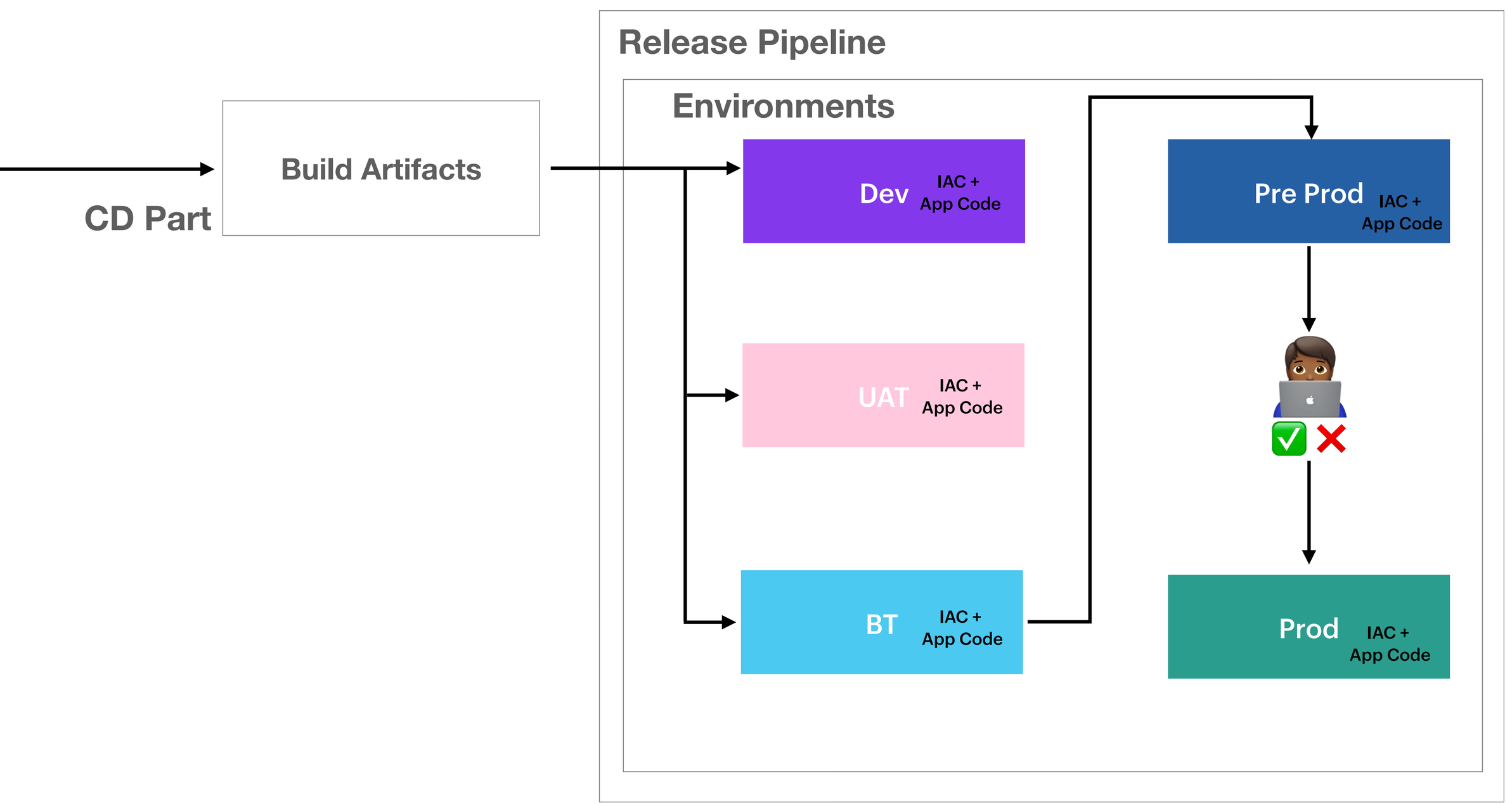

CODE
DEPLOY
BUILD
DEV
OPS
OPERATE
TEST
MONITOR

Agile Methodology

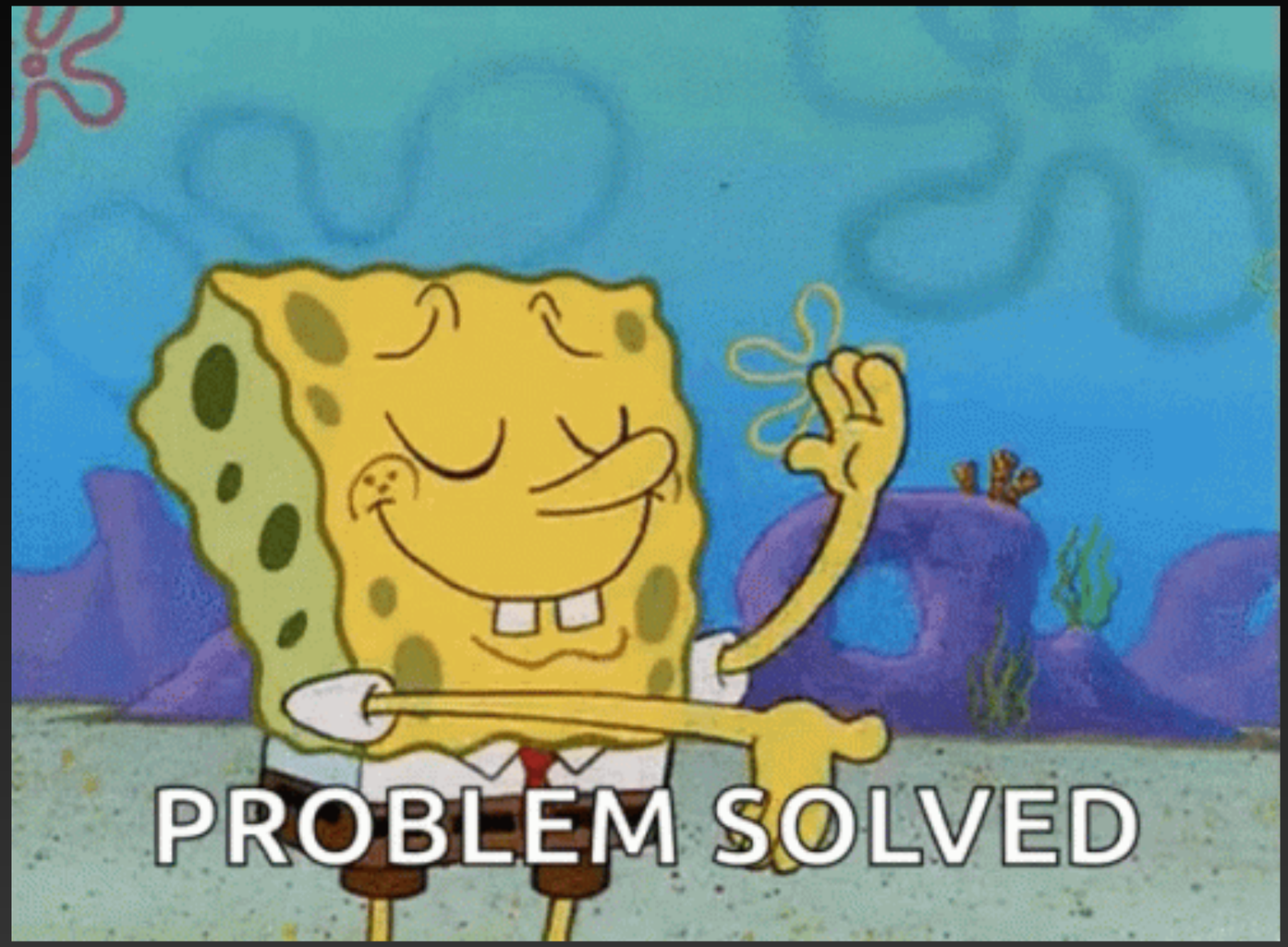


Continuous Integration


Developer
Writes Code
Frontend Dev
Backend Dev
CI tool
Build application From source code
Run Automated Tests
Everything Ok ?
CD Part
Problem detected
Changes Needed


Release Pipeline
Environments
CD Part
Build Artifacts
Dev
IAC +
App Code
UAT
IAC +
App Code
BT
IAC +
App Code
Pre Prod
IAC +
App Code
Prod
IAC +
App Code

Problems solved by DevOps



01

Bringing value to customers with faster production cycles

02

Increased productivity results in higher customer satisfaction

03

Saving costs through faster time to market

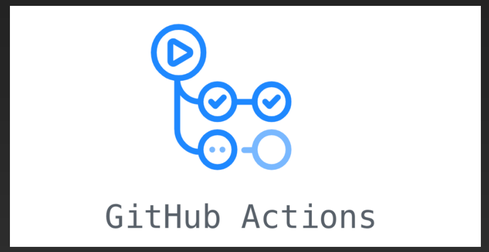
04

Enhanced quality with test automation

05

Improving problem resolution

DevOps Tools



many more ...

DevOps Periodic Table

The DevOps Burger !!

01. Learn Programming

- Python
- Golang
- JavaScript
- Ruby

02. Server Administration

- Linux
- Unix
- Windows

03. Network and Security

- TCP/IP Fundamentals
- Protocols: DNS, HTTP/s, FTP, SSL ....

04. Servers

Caching

- Redis
- MemCache

DataBase

NoSQL

- MongoDB
- Cassandra
- AWS DynamoDB
- Google DataStore

SQL

- OracleDB
- MySQL/ MariaDB
- PostgreSQL
- MS-SQL

Web Server

- Apache
- Nginx
- Tomcat
- IIS
- Jeffy

05. Infrastructure as Code

Configuration Management

- Ansible
- Puppet
- Chef
- Salt Stack

Container

- Docker
- rkt
- LXC

Container Orchestrators

- Kubernetes
- Openshift
- No Mad
- Docker Swarm

Infrastructure Provisioning

- Terraform
- AWS CloudFormation
- Azure Template
- Google Deployment Manager

06. CI/CD

- ArgoCD
- Teamcity
- Circle CI
- Jenkins
- AWS Code Pipeline
- Google Cloud Build
- Gitlab CI
- Bucket Pipeline
- Github Action

07. Monitoring and Logging

Infrastructure Provisioning

- Zabbix
- Promethues
- Grafana
- DataDog
- New Relix
- CheckMk

Logging

- ELK
- Graylog
- Splunk

08. Cloud

- AWS
- Azure
- GCP
- Openstack
- Alicloud
- IBM Bluemix

Color Representation

- Very Important
- Important
- Normal

Requirements

- Azure cloud knowledge
- Basic knowledge on build steps for programming language

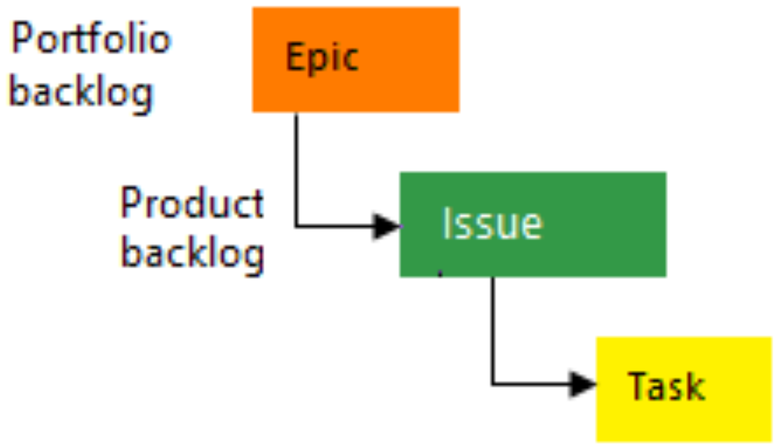

Azure DevOps

Azure Boards

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

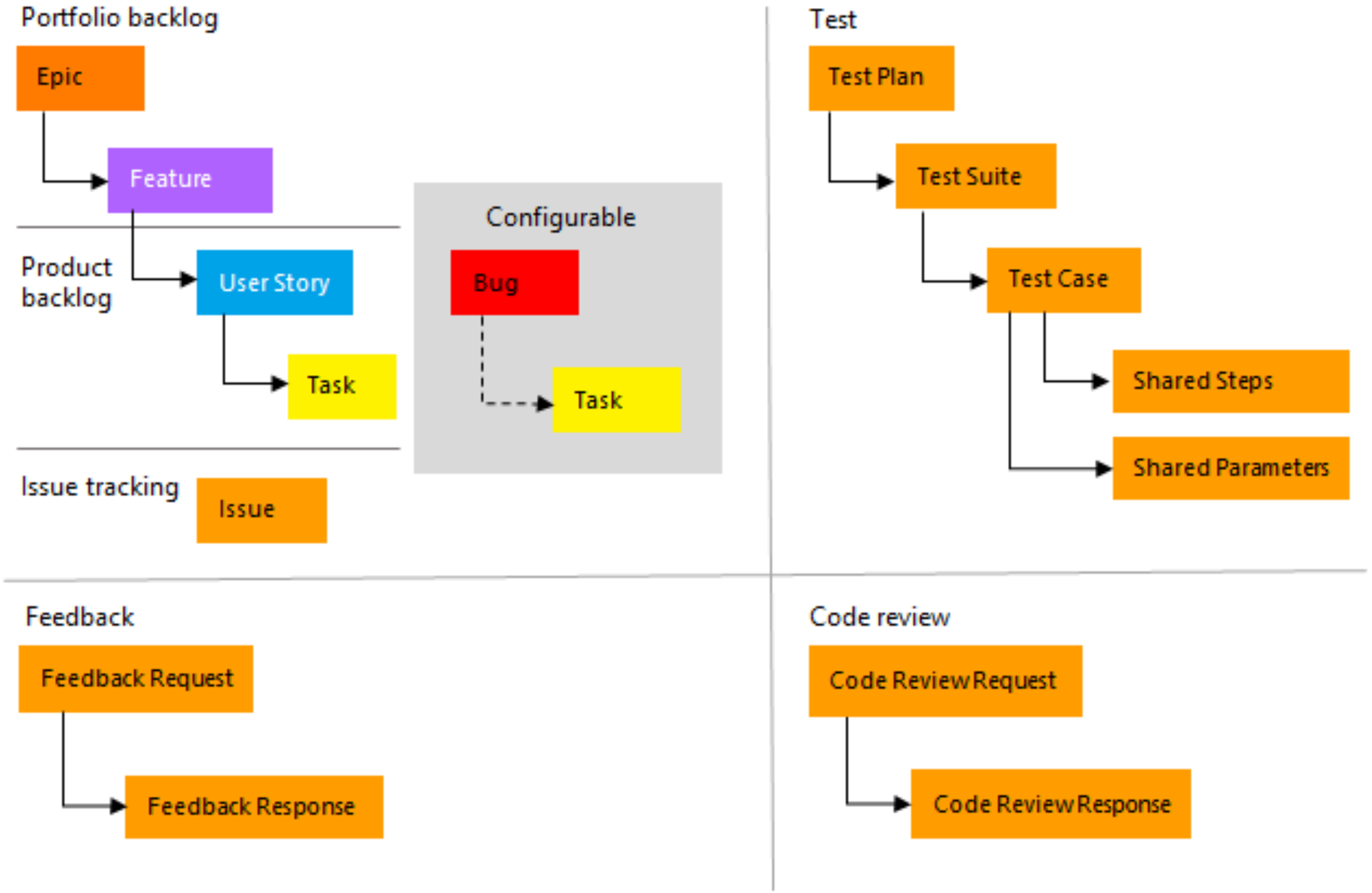

Basic
Agile
Scrum
CMMI

Basic Process

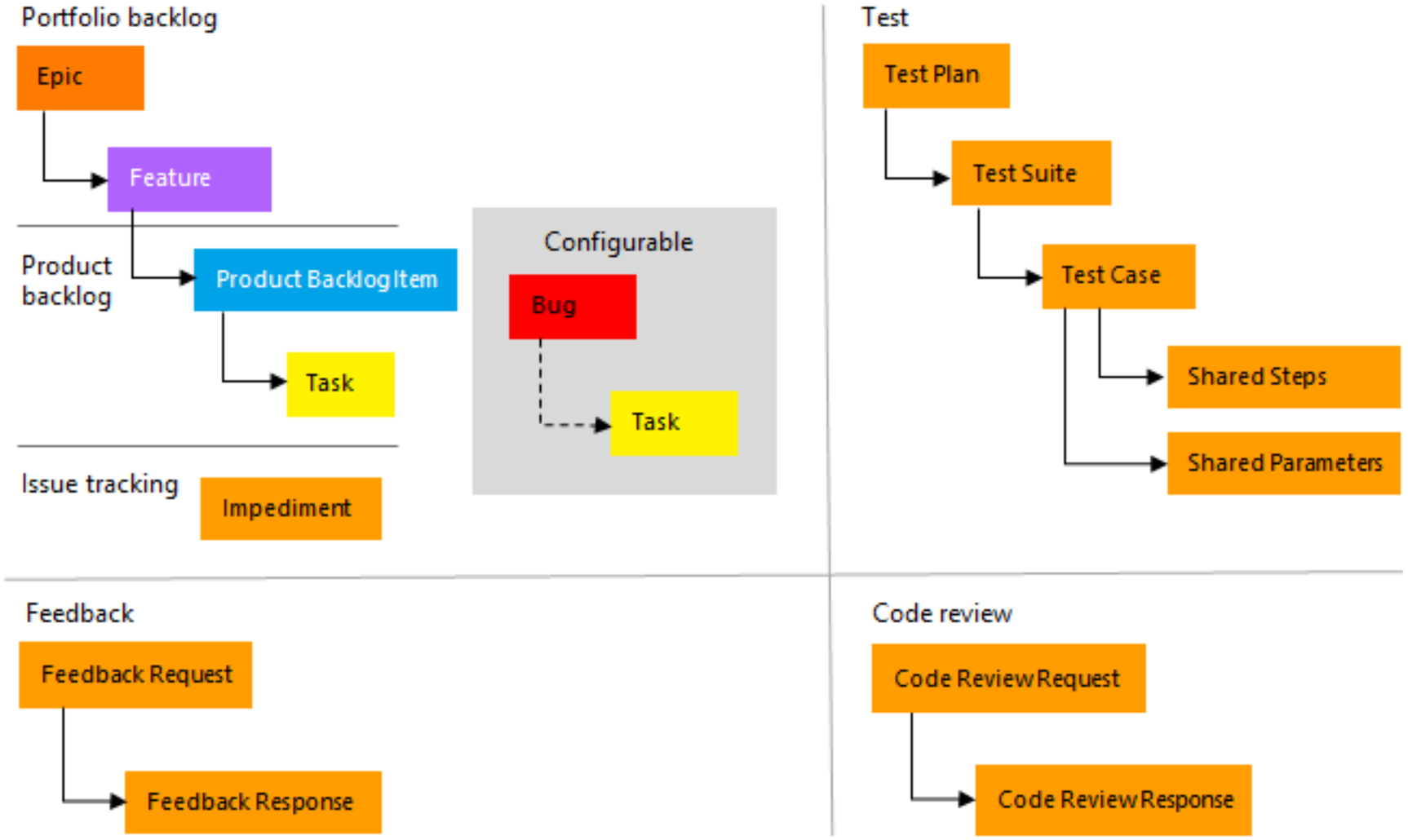


State	Title
To Do	Web site updates
Doing	Secure sign-in
Doing	Hello World web site
To Do	Design welcome screen
To Do	Standarize form factors
To Do	Change background color
To Do	About screen
To Do	Welcome back page
To Do	Change initial view
To Do	Service status
Doing	Resolve service status issues
Doing	Check performance
To Do	Change new item

Agile Process



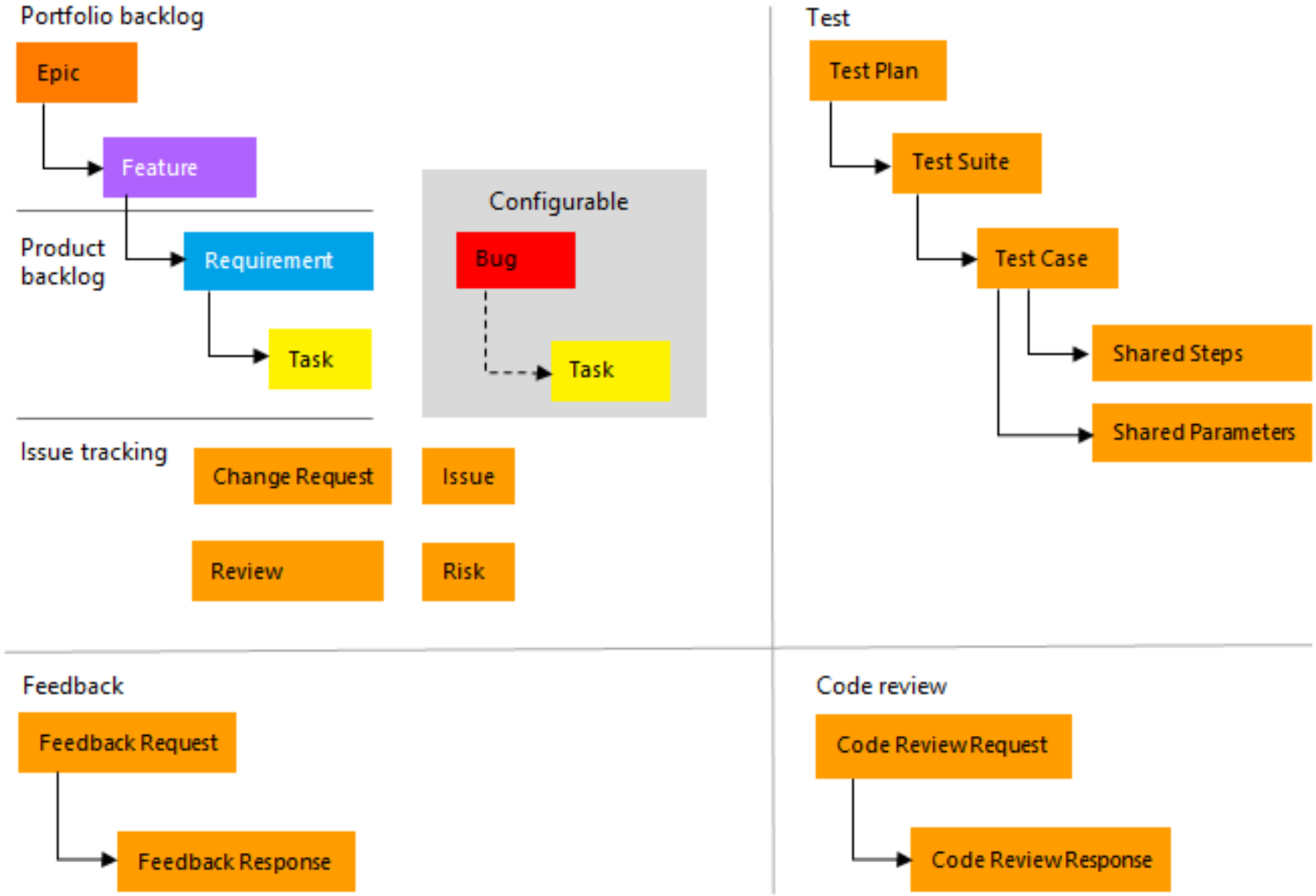
Scrum Process



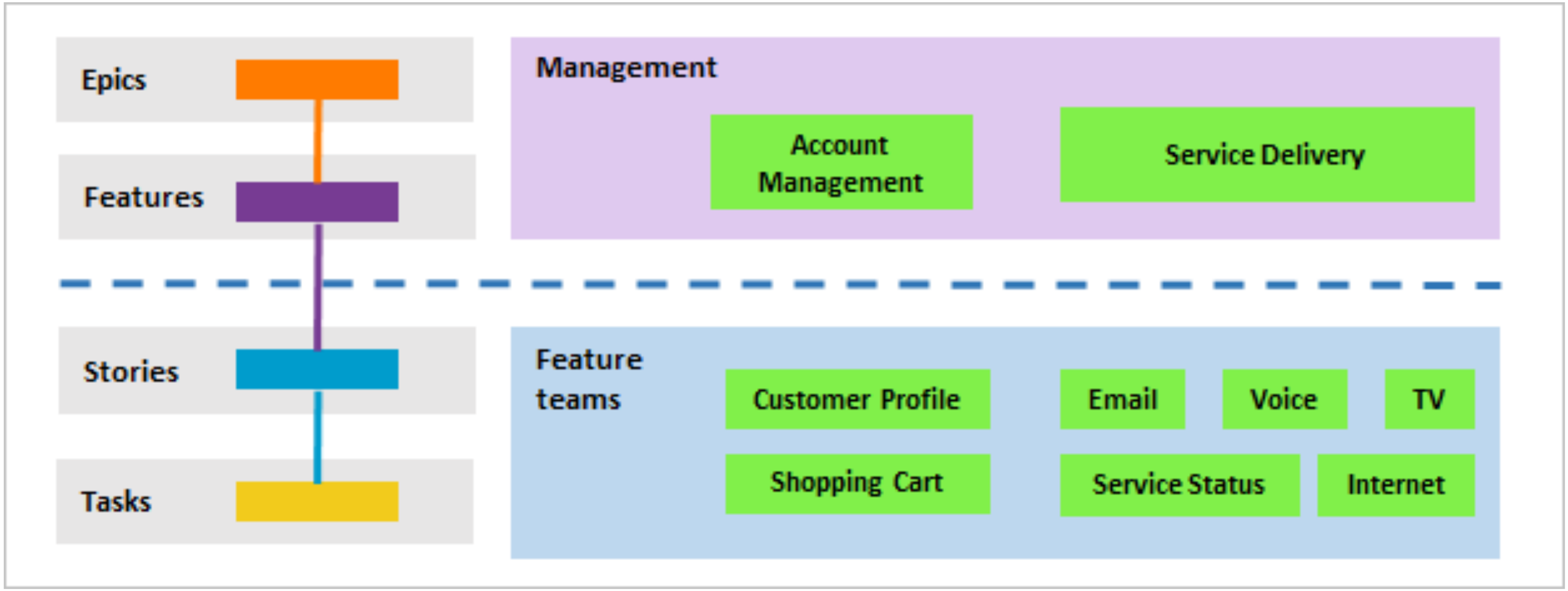
When to use Agile or Scrum

- Agile means “incremental, allowing teams to develop projects in small increments.
- Scrum is one of the many types of agile methodology
- Scrum is an iterative ongoing cycle of designing, developing, testing, rinse and repeat. Which include sprints
- agile emphasizes time and supporting team members to get X number of tasks completed by the end of the designated period of time.

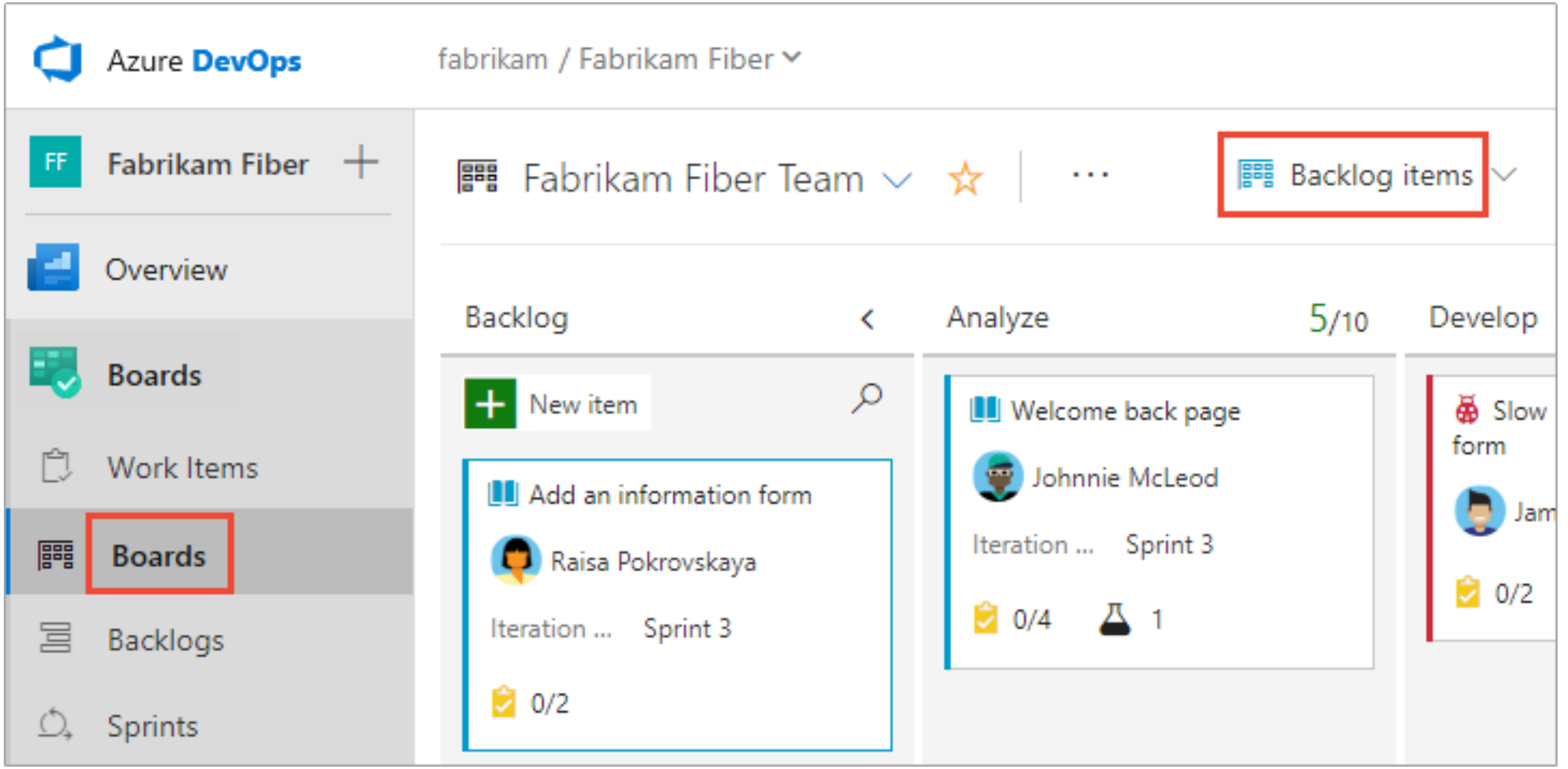
CMMI Process (Capability Maturity Model Integration)



Teams hierarchy



Kanban board

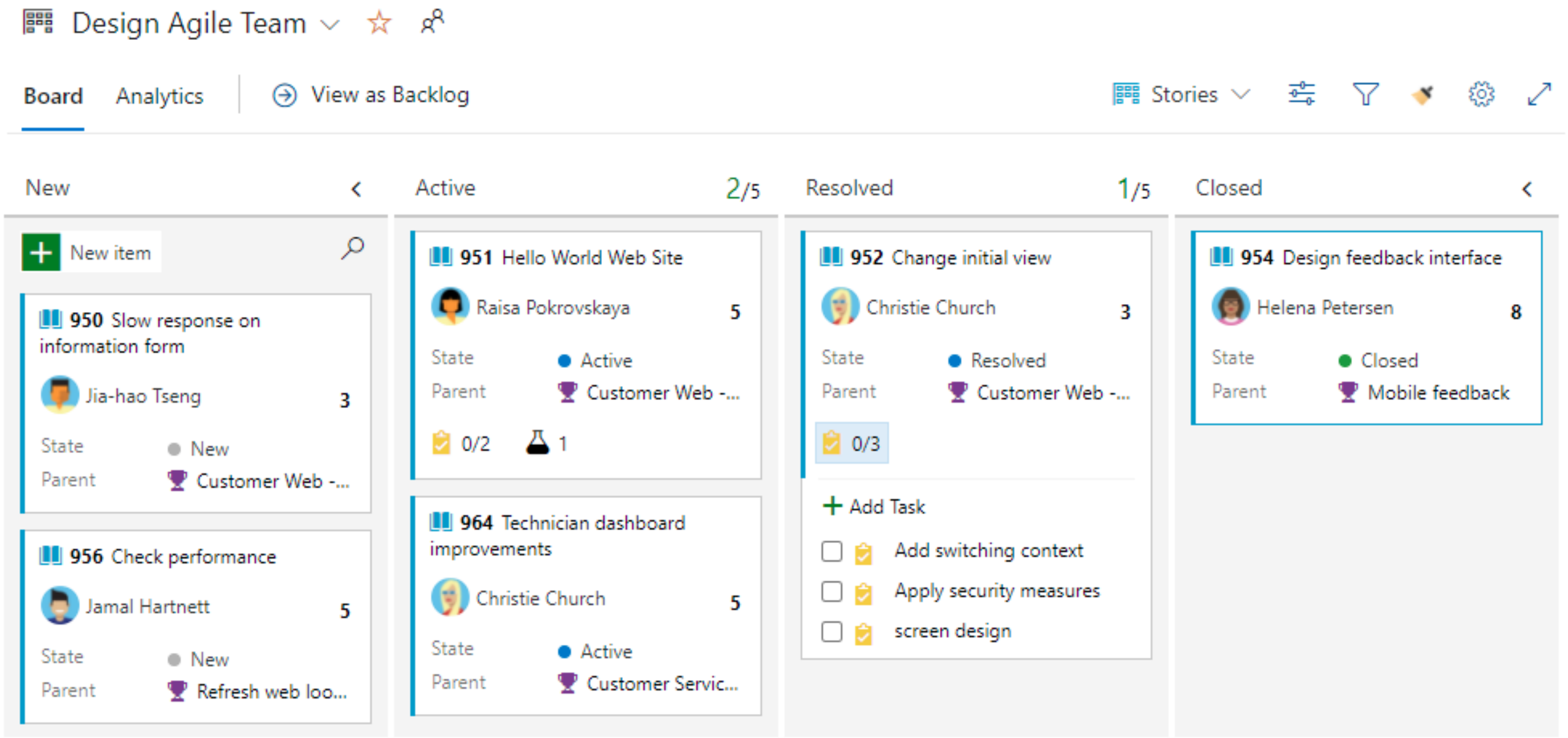


- **Product backlog**: By default lists User Stories (Agile), Issues (Basic), Product Backlog Items and Bugs (Scrum), or Requirements (CMMI). Provides options to show **Parents**, **Forecast**, and **In Progress** or **Completed** child items.

- **Portfolio backlog**: By default lists Features (all process models) for the Features backlog, and Epics (Agile, Scrum, and CMMI) for the Epic backlog. Provides options to show **Parents** and **In Progress** or **Completed** child items.

- **Sprint backlog**: By default lists all product backlog items assigned to the selected iteration, regardless of status. Provides options to show **Work details**.

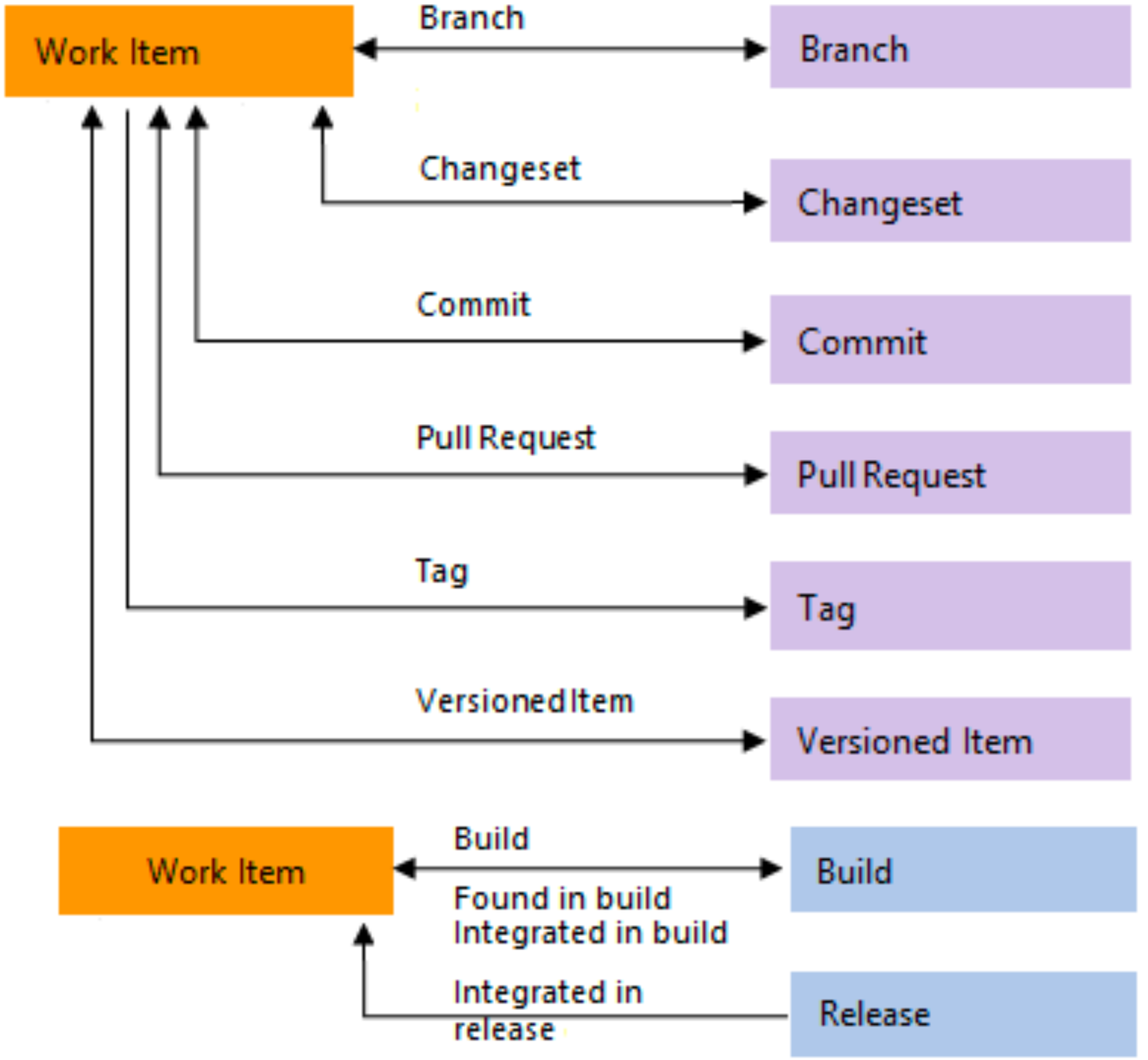
Boards walkthrough



GitHub to Azure Boards work items

- Connect Azure Boards to GitHub :: Projects —> GitHub connections
- AB#{ID}
- Fixed AB#123
- Fixes AB#123, AB#124, and AB#126
- Fixing multiple bugs: issue #123 and user story AB#234
- **Add link** under the Development section - PR, Commit, issue

- Capture a work item as a template
- Manage work item templates
- Boards configurations under project settings
- Add a custom field to a work item type (Inheritance process)
- https://learn.microsoft.com/en-us/azure/devops/organizations/settings/work/add-custom-field?view=azure-devops
- You can apply the new process to existing project
- Use Excel to create all types of workobjects (only windows)
- https://learn.microsoft.com/en-us/azure/devops/boards/backlogs/office/bulk-add-modify-work-items-excel?view=azure-devops&tabs=agile-process
- Bulk import using csv (Mac/ windows)


Work Item
Branch
Branch
Changeset
Changeset
Commit
Commit
Pull Request
Pull Request
Tag
Tag
VersionedItem
Versioned Item
Work Item
Build
Build
Found in build
Integrated in build
Integrated in release
Release

Keyboard shortcuts

- Keyboard shortcuts
- ? - to show all keyboard shortcuts

Exercises for Section 1 Part 1

- Create a agile Project , write a query to retrieve bugs
- integrate github with Azure boards, fix an issue from commit

Section 1 part 2

Some important metrics

- Burndown chart - begin with the total amount of planned work and then as work is completed graphs the remaining work.
- Velocity and velocity chart - Velocity provides a useful metric for gaining insight into how much work your team can complete during a sprint cycle.
- Burnup chart - Burnup charts track work as it is completed over time. They are useful to show the rate at which work is getting completed
- Lead Time - Lead time measures the total time elapsed from the creation of work items to their completion
- Cycle time - measures the time it takes for your team to complete work items once they begin actively working on them
- Cumulative flow diagram

Scrum best practices

- sprint planning meetings
- Set sprint goals
- Manage your technical debt
- Address any known bug debt every sprint
- Role of the Scrum Master
- Daily Scrum meetings
- Sprint review meetings
- Sprint retrospective meetings

Wikis walkthrough

Sprint backlog capacity

Work details capacity View

Custom dashboards using powerBI

Exercises for Section 1 Part 2

- Configure dashboard and add all chart types
- Assign work items in sprint and capacity planning

Section 2 part 1

Azure Repos

- Git: distributed version control
- Team Foundation Version Control (TFVC): centralized version control

Azure Repos Key concepts

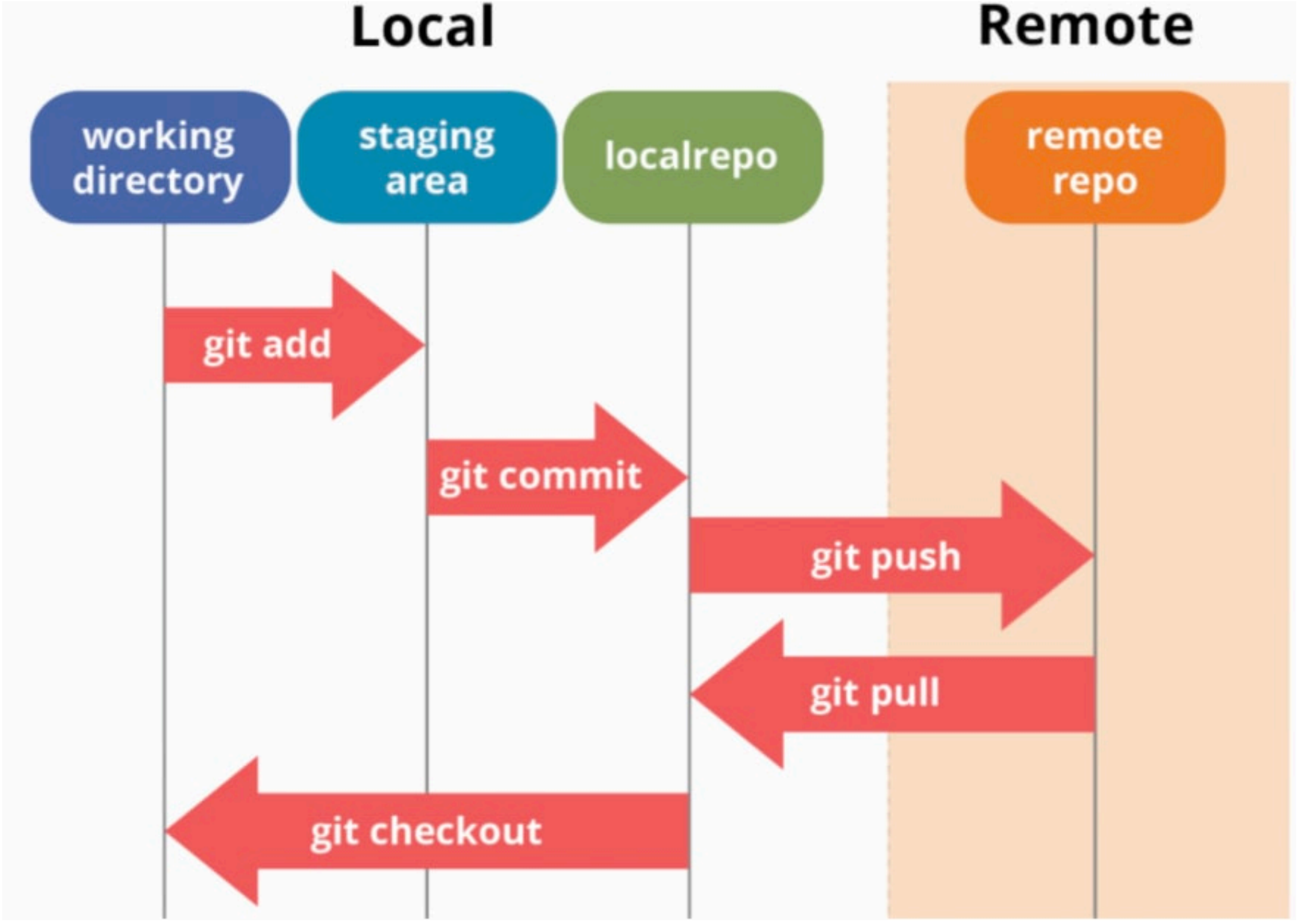
- Branch
- Branch policies
- Clone
- Commit
- Fork
- Git workflow
- Git pull
- Git push
- Branch merge
- Pull request
- Ignore files
- Git config
- Git branching strategy
- Branch permissions
- Merge types
- Lock branch
- Merge conflicts
- Cherry-pick

Git config

- git config --global user.name "FIRST_NAME LAST_NAME”
- git config --global user.email “MY_NAME@example.com"
- git remote add origin <url>

```
git remote remove origin
```

Git commit Flow



Exercises for Section 2 part 1

- Create a repo in azure Repos and push code
- Create a new develop branch from master
- Create and resolve conflicts in branches (learn)

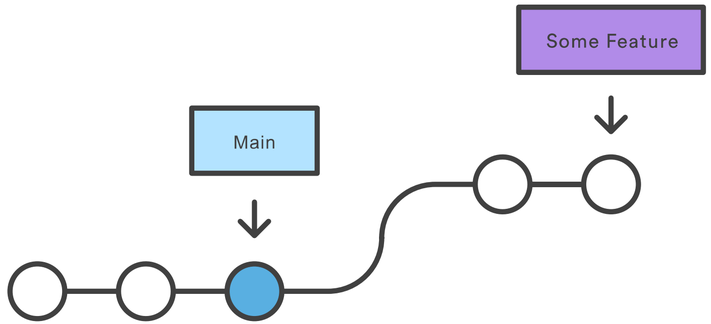
Section 2 part 2

Merge Types in Azure DevOps

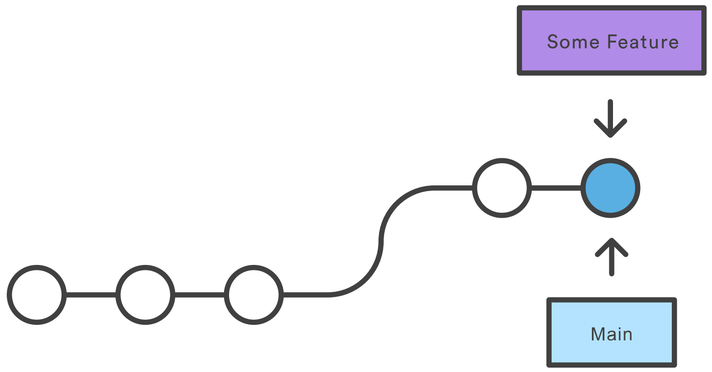
- Merge (no fast forward)
- Squash commit
- Rebase and fast forward
- Semi-linear merge

Fast Forward Merge

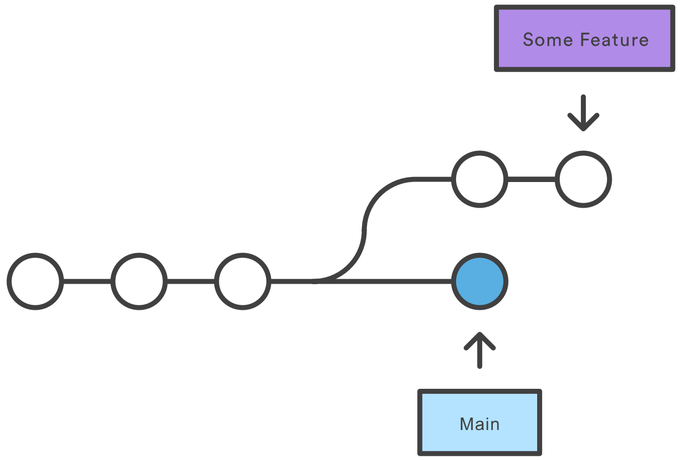
Before Merging



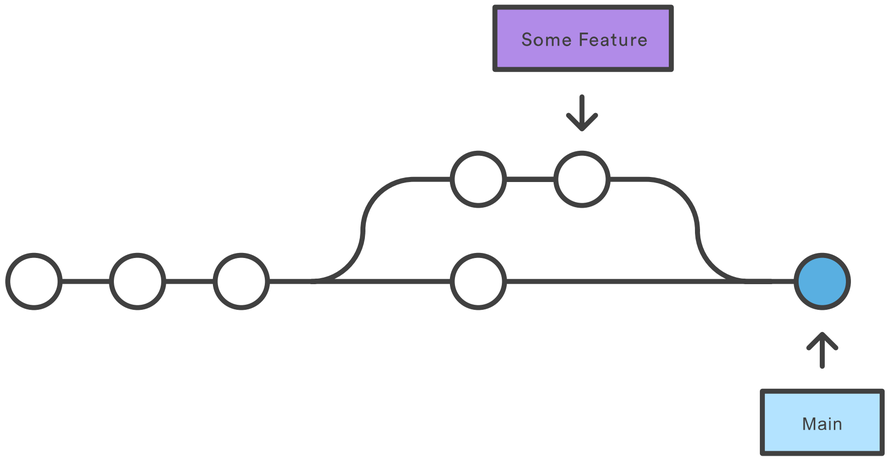
After a Fast-Forward Merge



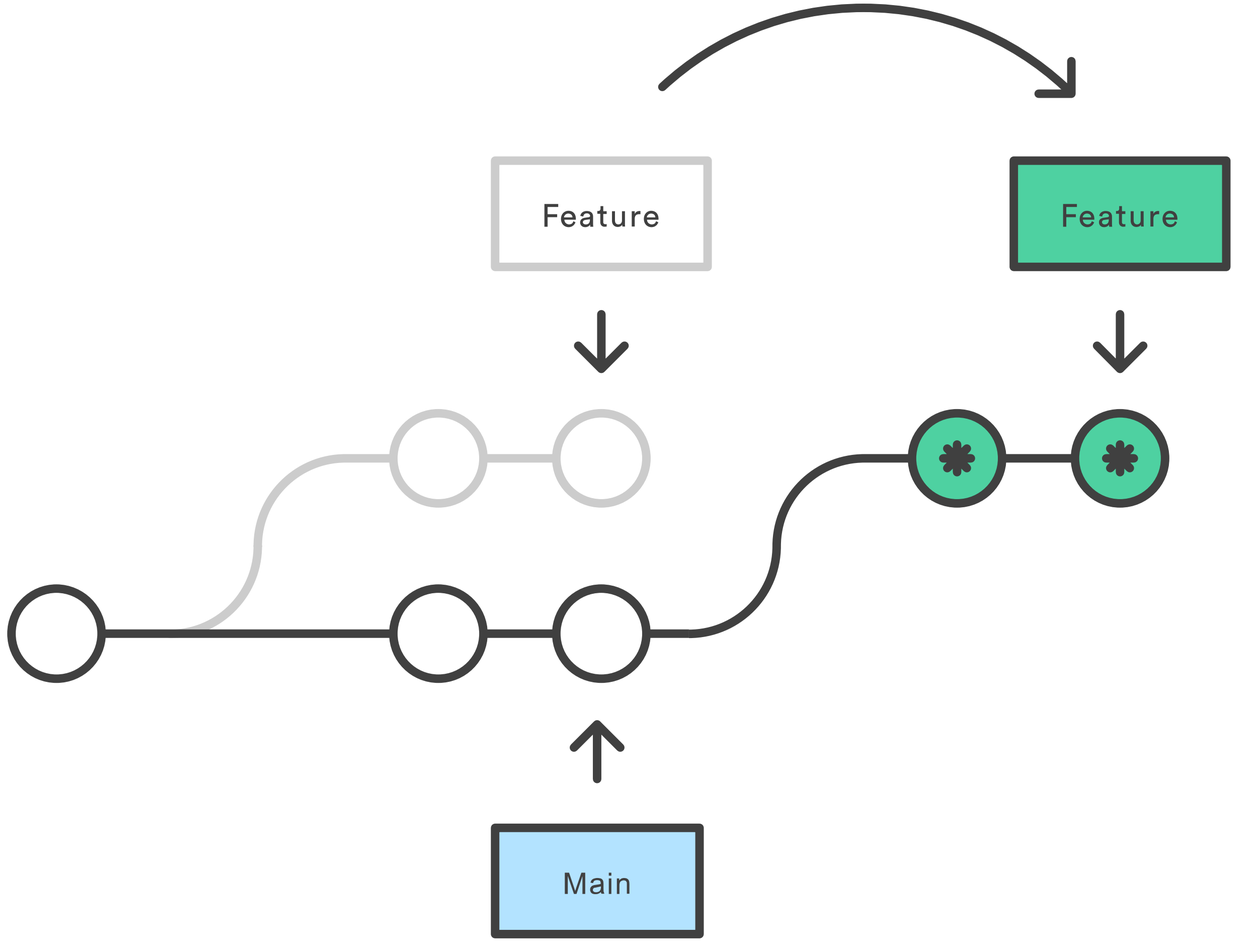
Before Merging



After a 3-way Merge



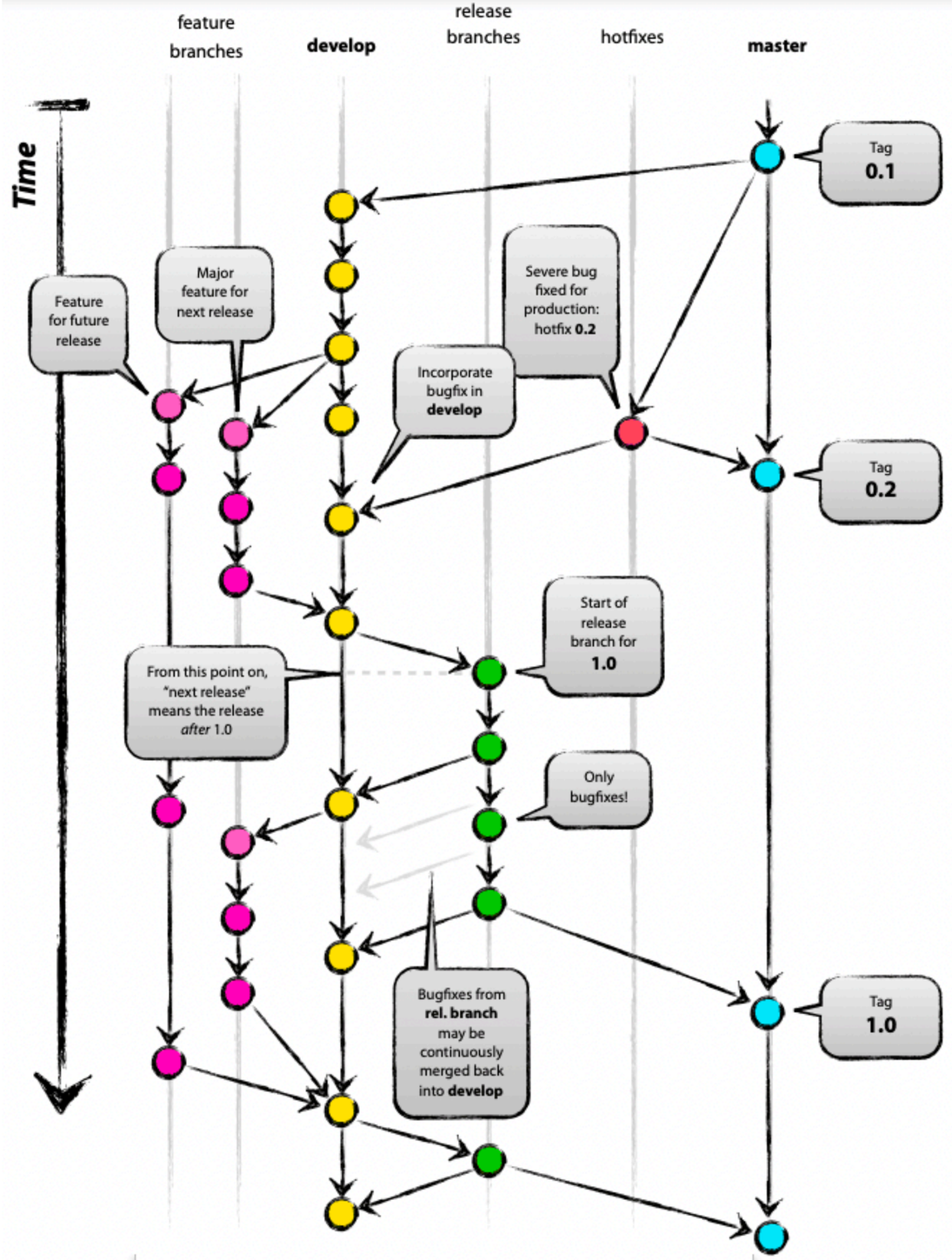
Git rebase



Brand New Commits

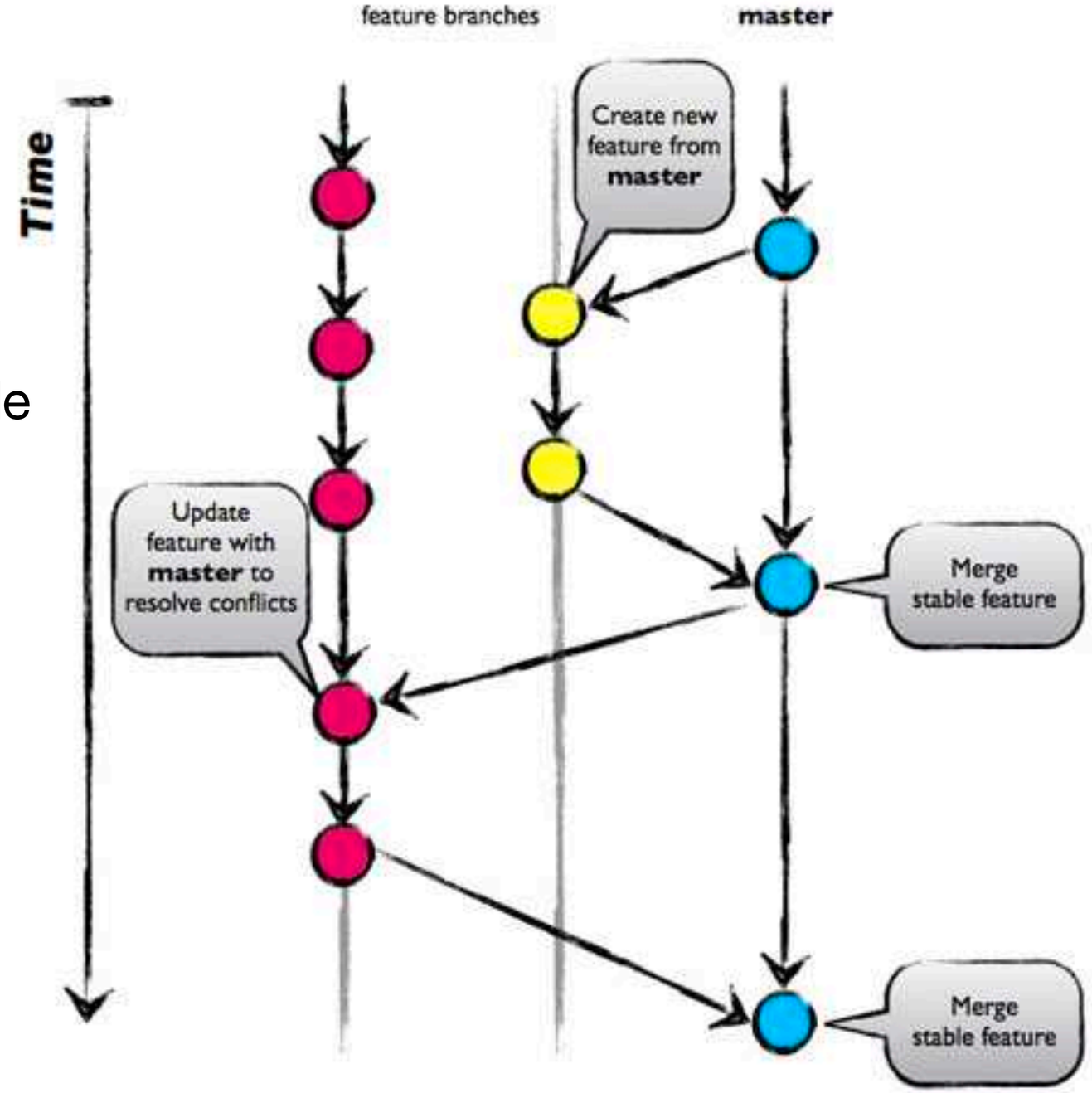
Git Flow

- Master, develop - infinite lifetime
- Feature - created on top of develop
- Release - prod release , develop merged back to develop and master
- Hot-fix - quick prod release , bug
- Remaining branches are short lived branches



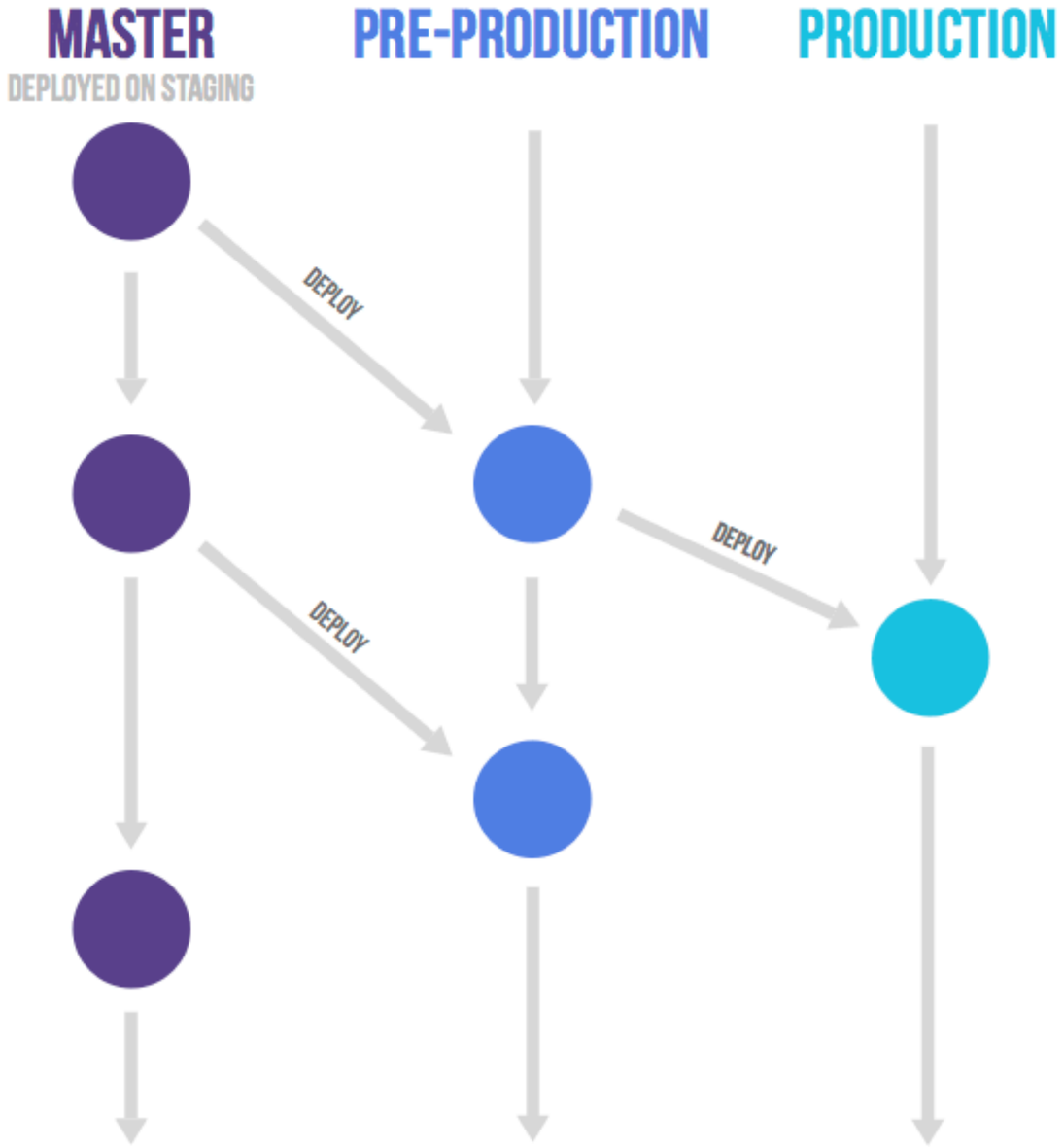
GitHub Flow

- Feature branches from master
- master code in a constant deployable state
- support continuous integration and continuous delivery processes.
- Focus on agile method



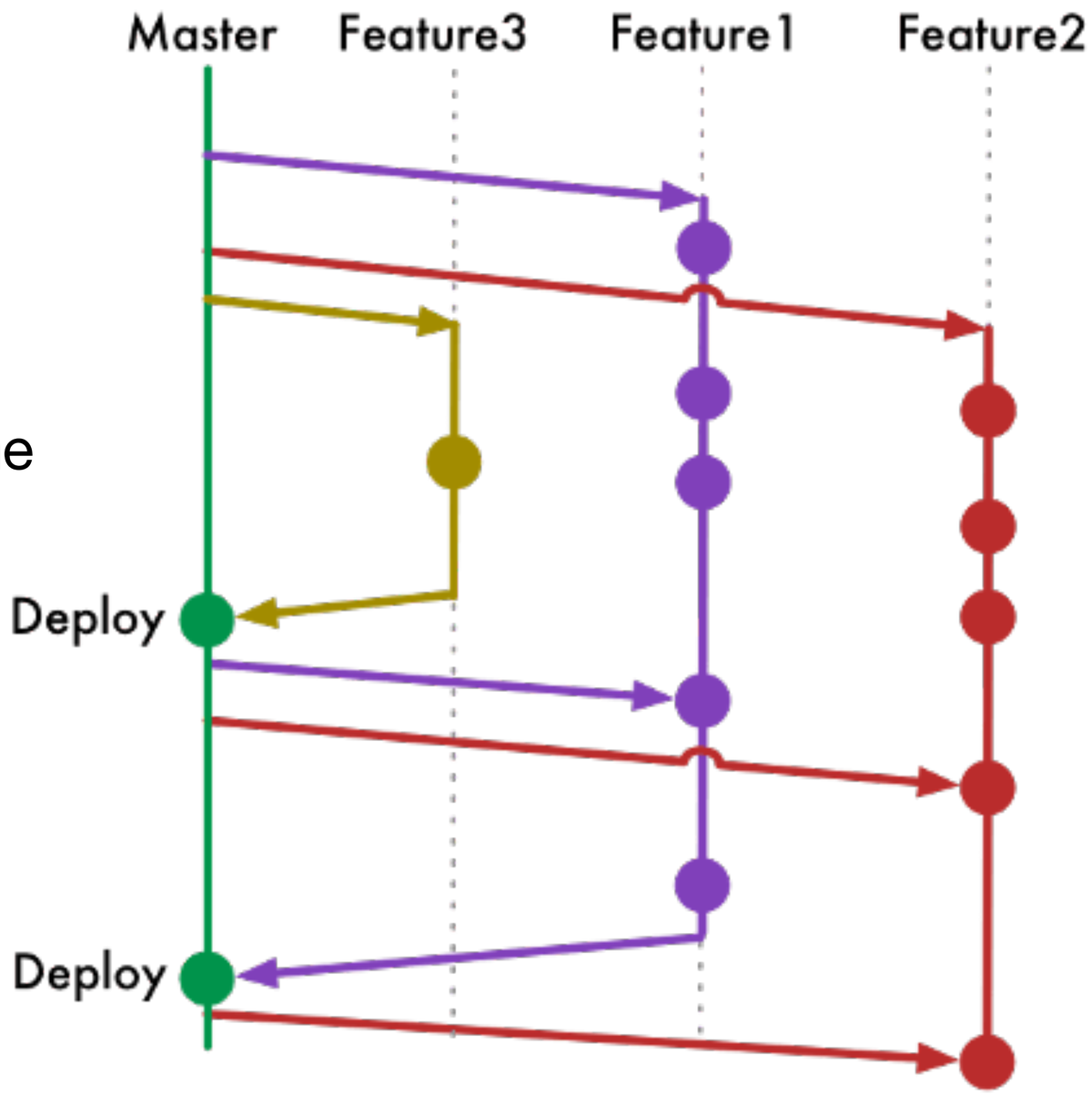
GitLab Flow

- Feature branches from master
- Multiple version of code in different environments
- Resolve the issue before going to production



Trunk-based Development

- Requires no branches
- Code changes into shared trunk
- This trunk can be released at anytime
- All developers work on the same branch.
- Used for feature flags



Best branching strategy for your team

Product type and its release method	Team size	Collaboration maturity	Applicable mainstream branch mode
All	Small team	High	Trunk-Based Development (TBD)
Products that support continuous deployment and release, such as SaaS products	Middle	Moderate	GitHub-Flow and TBD
Products with a definite release window and a periodic version release cadence, such as iOS apps	Middle	Moderate	Git-Flow and GitLab-Flow with release branch
Products that are demanding for product quality and support continuous deployment and release, such as basic platform products	Middle	Moderate	GitLab-Flow
Products that are demanding for product quality and have a long maintenance cycle for released versions, such as 2B basic platform products	Large	Moderate	Git-Flow

Project settings -> repositories walkthrough

- Permissions for different groups

Repos -> branches -> 3 dot on branch -> branch policies

- This will give build validation , status checks , approvals

*These concepts can be explained after pipeline creation, dependency on build pipeline

Cross service integration

- End-to-end traceability
- Traceability and linking
- Branch from a requirement
- Create a pull request from a requirement
- Add and run tests from requirements
- Deployment and Development controls
- Bug traceability

Exercises for Section 2 part 2

- Merge a branch using squash merge with PR
- Merge a branch using rebase with some commits ahead
- Create branch policies for main branches
- Build validation , status checks
- Add reviewers in PR , link work items at all stages commit, pr , merge
- Lock a branch

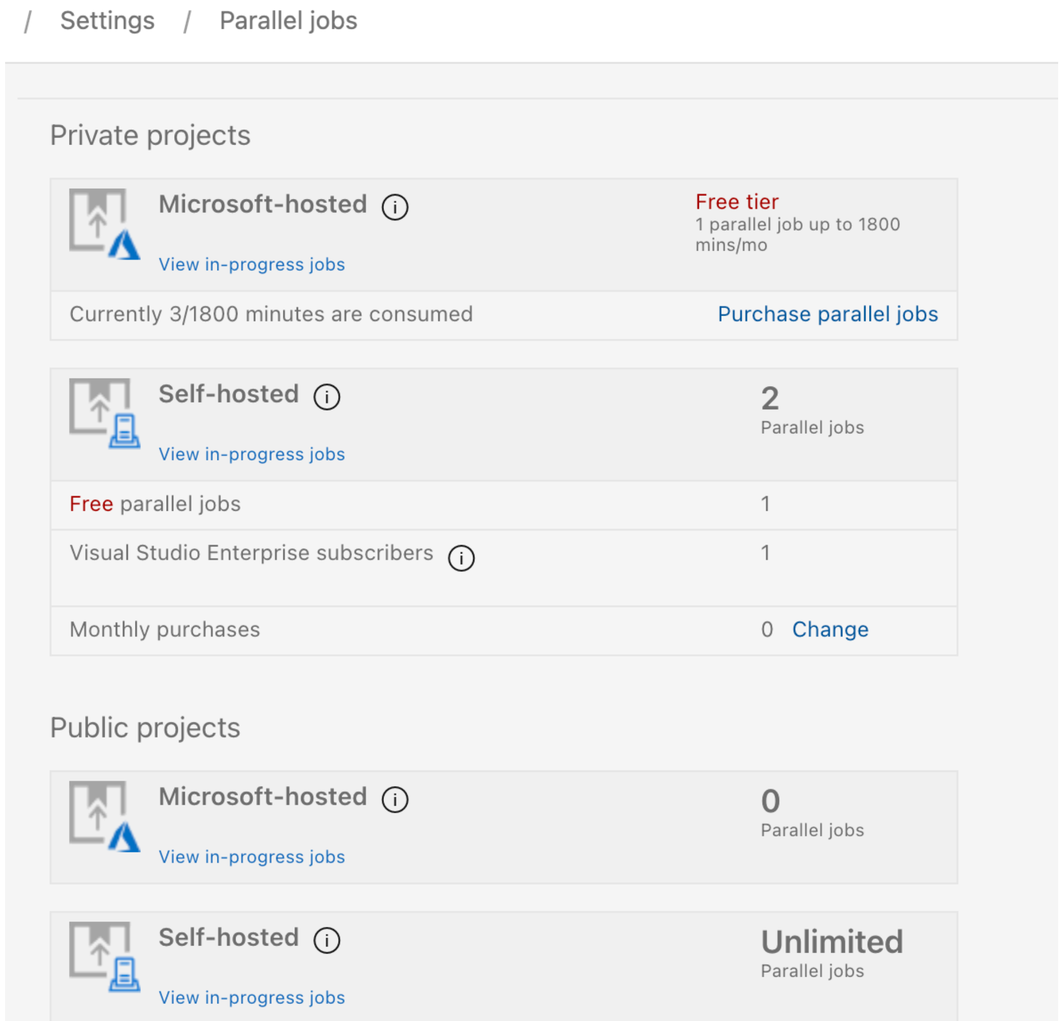
Section 3

Azure Pipelines

- Continuous Integration
- Continuous Delivery
- Continuous Testing
- Azure DevOps Services
- Azure DevOps Server

Free grant of parallel jobs

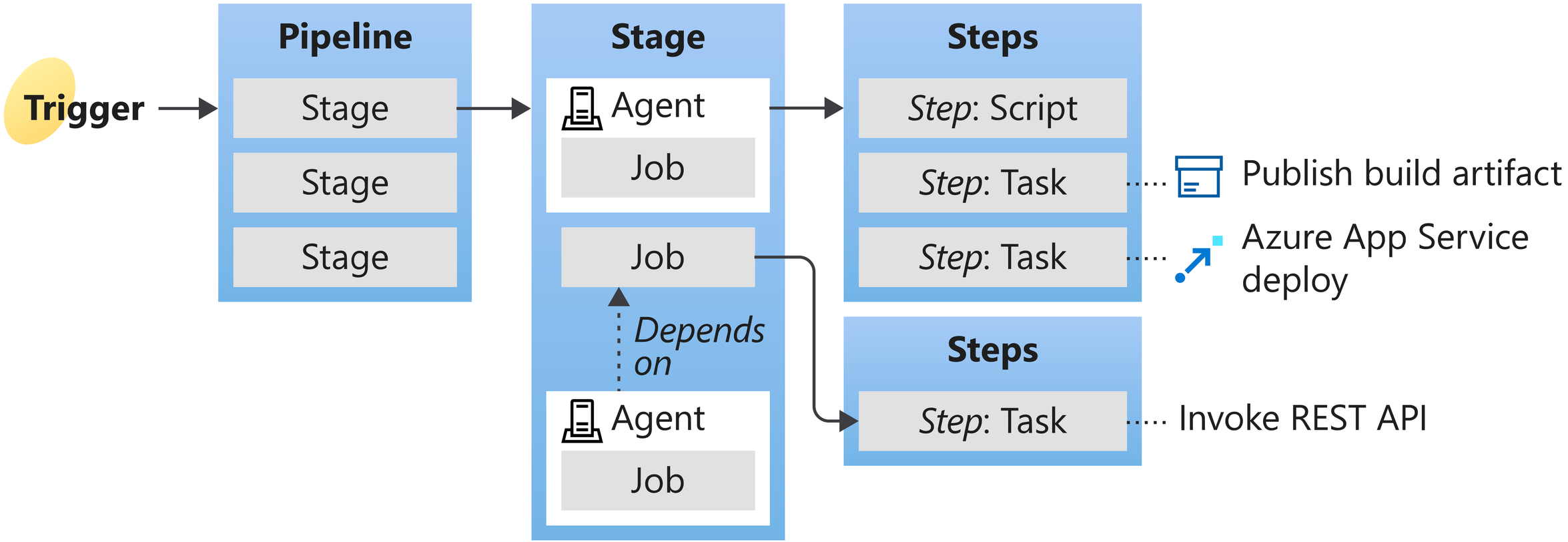
- https://aka.ms/azpipelines-parallelism-request



Key concepts

- Agent
- Approvals , checks , gates
- Artifact
- Continuous delivery
- Continuous integration
- Deployment
- Deployment group
- Environment
- Agent
- Job
- Pipeline
- Release
- Run
- Script
- Stage
- Step
- Task
- Triggers
- Library , variables
- Taskgroup
- Templates
- Caching
- Tests
- Flaky tests

Key concepts overview



Feature Availability

- https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/pipelines-get-started?view=azure-devops#feature-availability

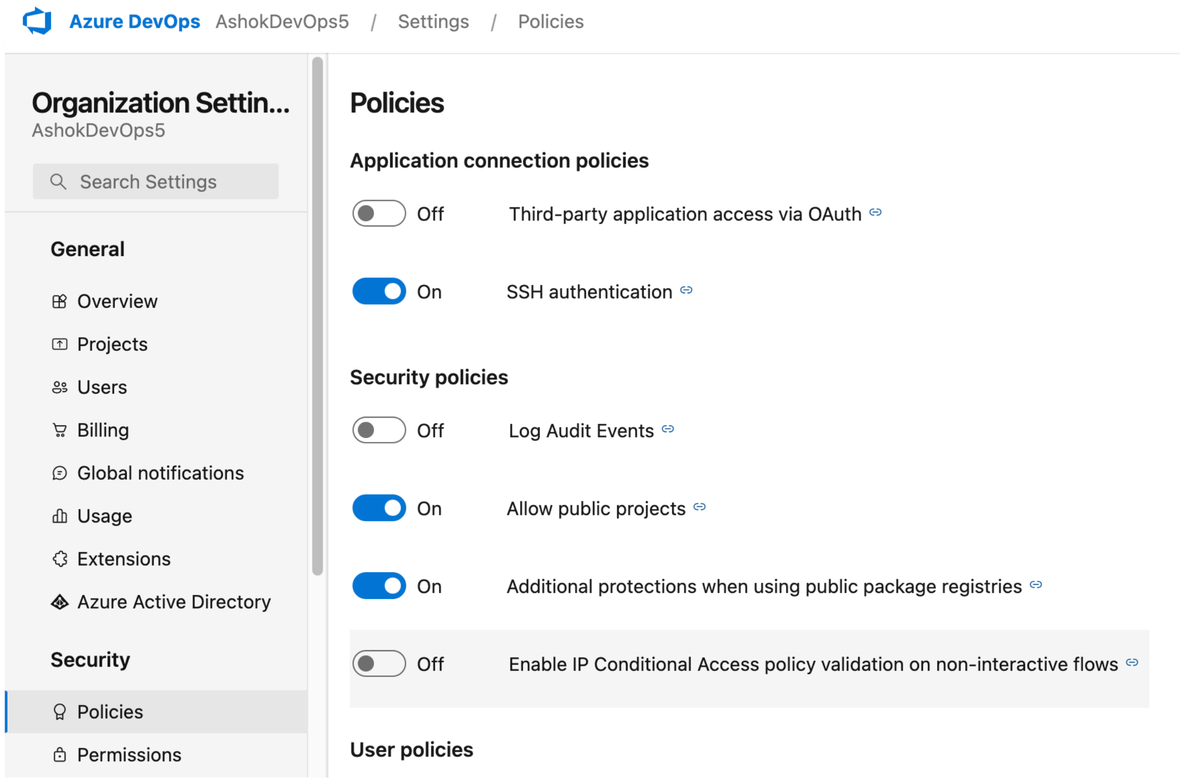
Self Hosted agent setup

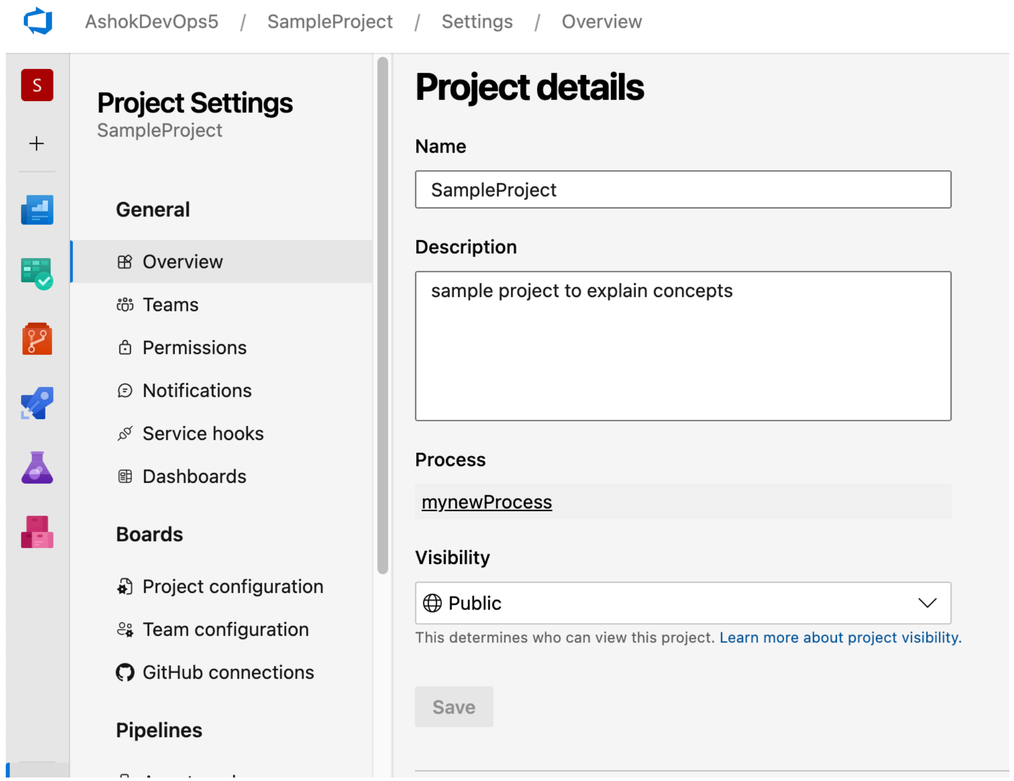
- PAT
- VM creation
- Git, .net
- Post vm steps
- Install visual studio

YAML concepts

- ———…
- Key-value pairs
- List
- Dictionary
- Comments

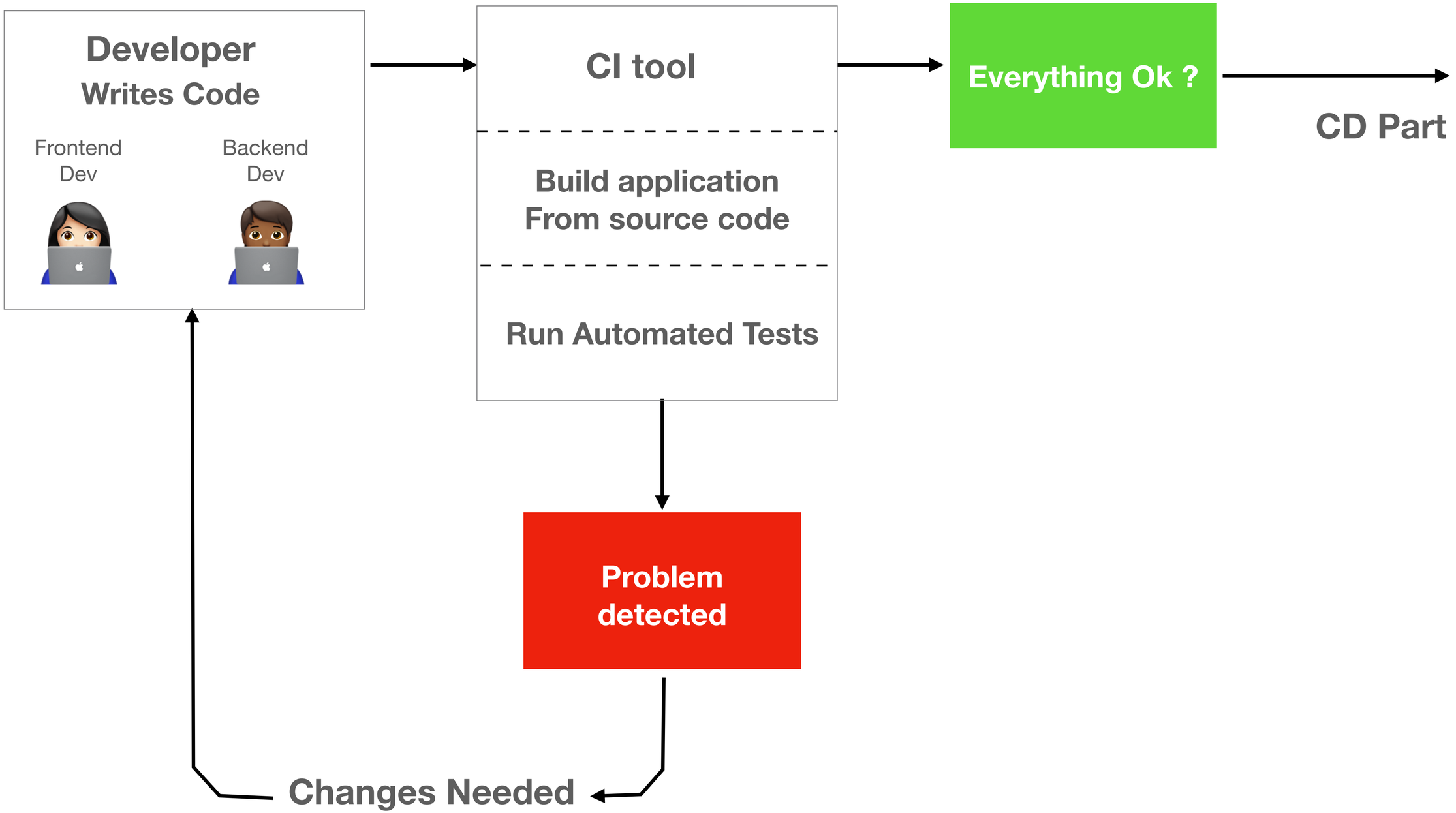
Change project private to public





Build Pipeline Creation

Continuos integration

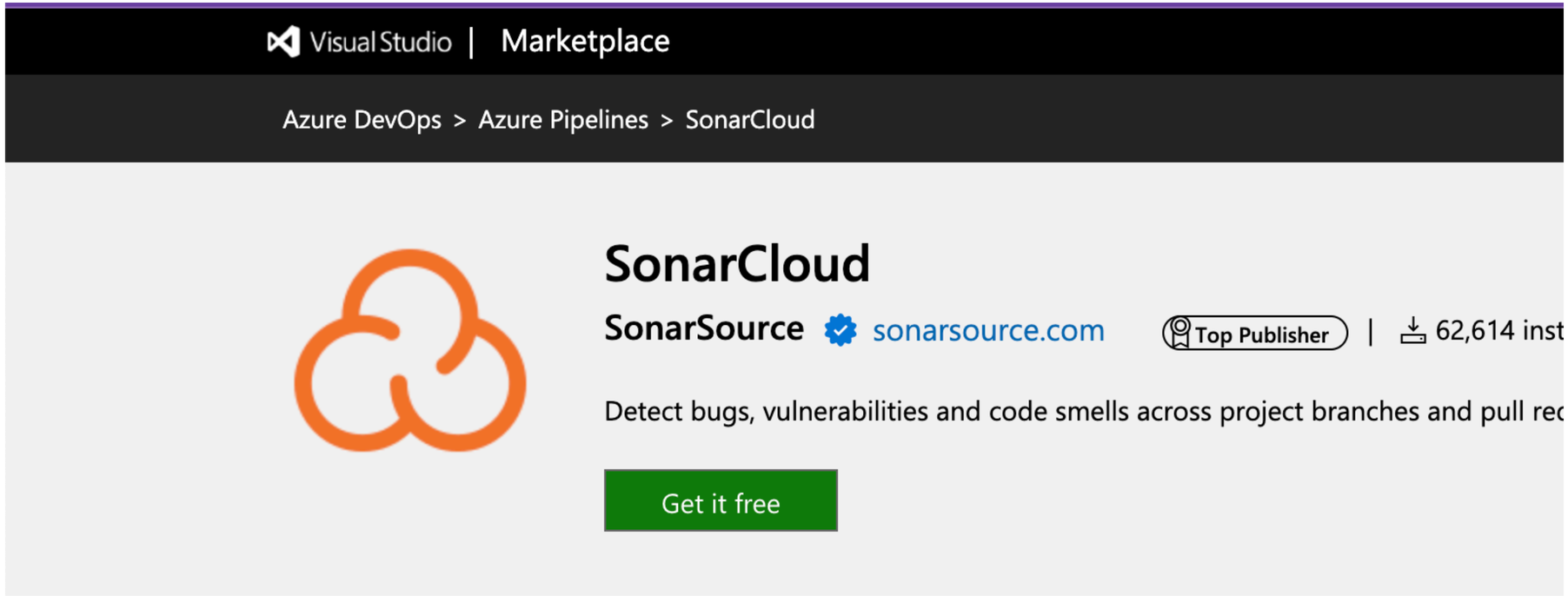


Sample pipeline creation in Azure Devops

- Pre-defined variables in pipelines
- https://learn.microsoft.com/en-us/azure/devops/pipelines/build/variables?view=azure-devops&tabs=yaml

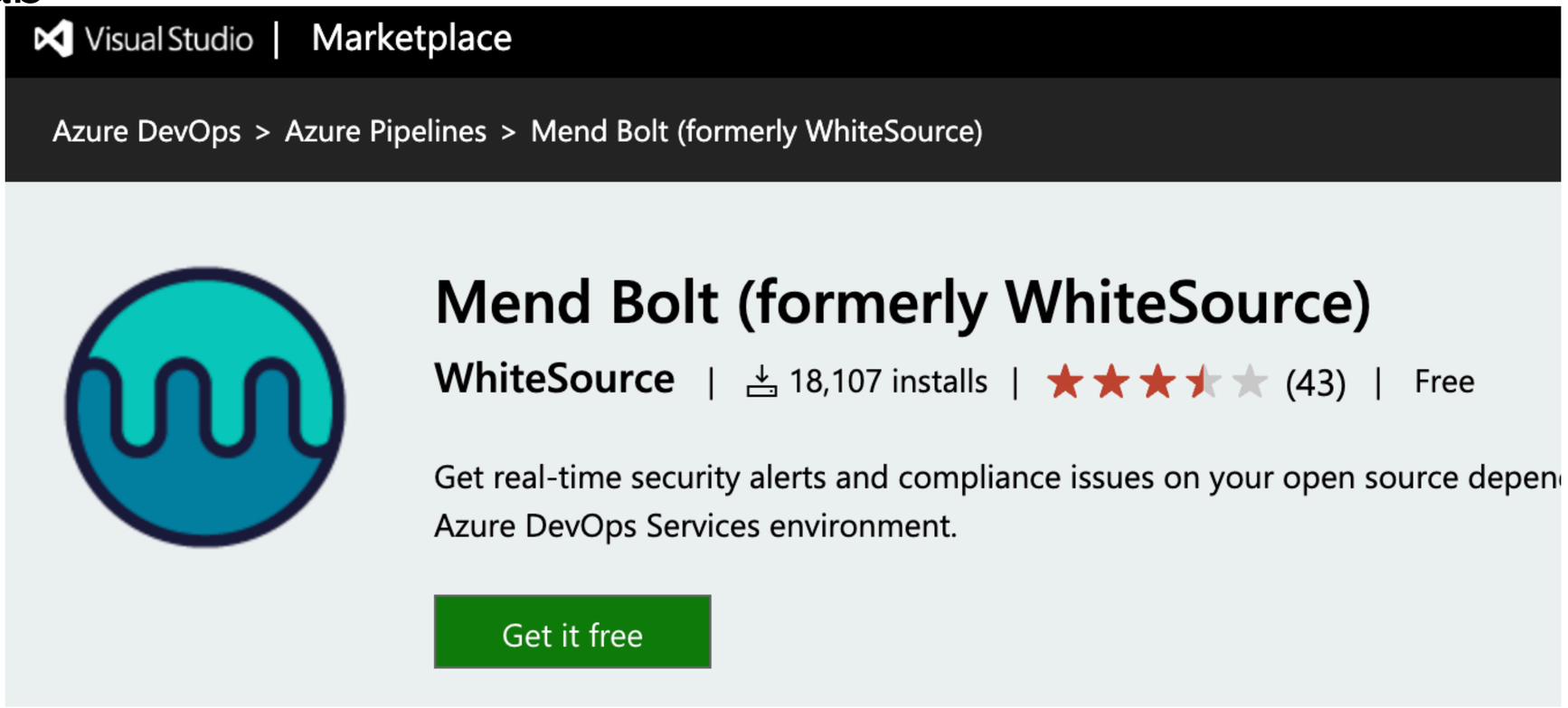
Sonar cloud

- Code definition
- Quality Gates (pass /failed)



Mendbolt

- FREE extension, which scans all your projects and detects open source components, their license and known vulnerabilities
- Mend tab



Create work item on failure

Triggers

- Commit
- PR
- Scheduled triggers - cron
- Branch considerations
- filters - include / exclude

Template usage

- template jobs
- Template with parameters
- Variables
- Secret variables

Task control options

```
- task: string  # reference to a task and version, e.g. "VSBuild@1"
  condition: expression     # see below
  continueOnError: boolean  # 'true' if future steps should run even if this step fails; defaults to 'false'
  enabled: boolean          # whether or not to run this step; defaults to 'true'
  retryCountOnTaskFailure: number # Max number of retries; default is zero
  timeoutInMinutes: string  # how long to wait before timing out the task
  target: string            # 'host' or the name of a container resource to target
```

Jobs

- Dependent jobs
- Conditions job
- Multi-job configs

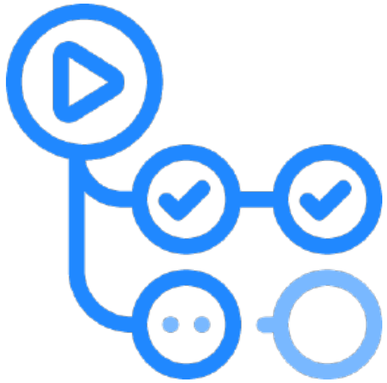
Deployment jobs

- special type - Deployment job
- Deployment history
- Apply deployment strategy
- runOnce
- rolling
- canary

- https://learn.microsoft.com/en-us/azure/devops/pipelines/process/deployment-jobs?view=azure-devops#schema

Deployment Job

- lifecycle hooks - preDeploy, deploy,
- routeTraffic,
- postRouteTraffic
- on: failure on: success



Github Actions

- Yaml
- Workflows
- Jobs
- Github Runners
- Self Hosted Runners
- WorkFlow Dispatch

Exercises for Section 3

- Create CI pipeline for favourite programming language - Microsoft Hosted
- Use self hosted for same CI pipeline
- Create multiple agents from single VM
- For CI , use YAML and Classic pipelines
- Integrate SonarQube , MendBolt
- Deploy app to Azure WebApp
- Include secrets in pipelines using key vault
- Implement CI/CD pipeline for app using Github Actions

Section 4

Azure Artifacts

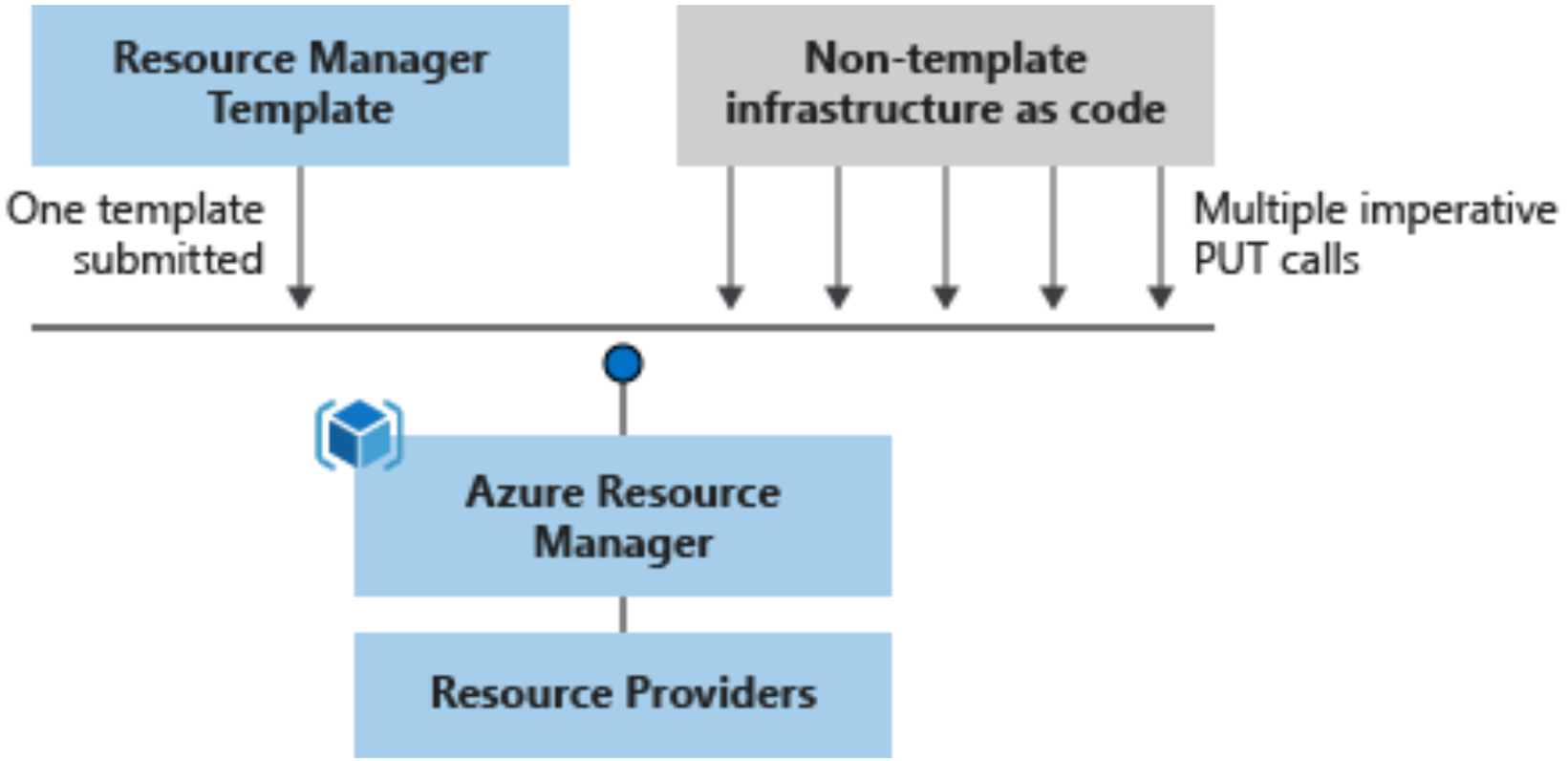
- Feeds - store, manage, and group your packages
- FeedViews - hare package versions that have been tested and validated but hold back on packages
- UpStream Sources - store packages from various sources in a single feed
- Azure Artifacts pricing

Azure Testplans

- Rich and powerful tools everyone in the team can use to drive quality and collaboration throughout the development process.
- Easy-to-use, browser-based test management solution provides all the capabilities required for planned manual testing
- Test plans Pricing

ARM Template structure

- $schema
- contentVersion
- parameters
- variables
- resources
- outputs



Infra as Code

Bicep

HashiCorp
Terraform

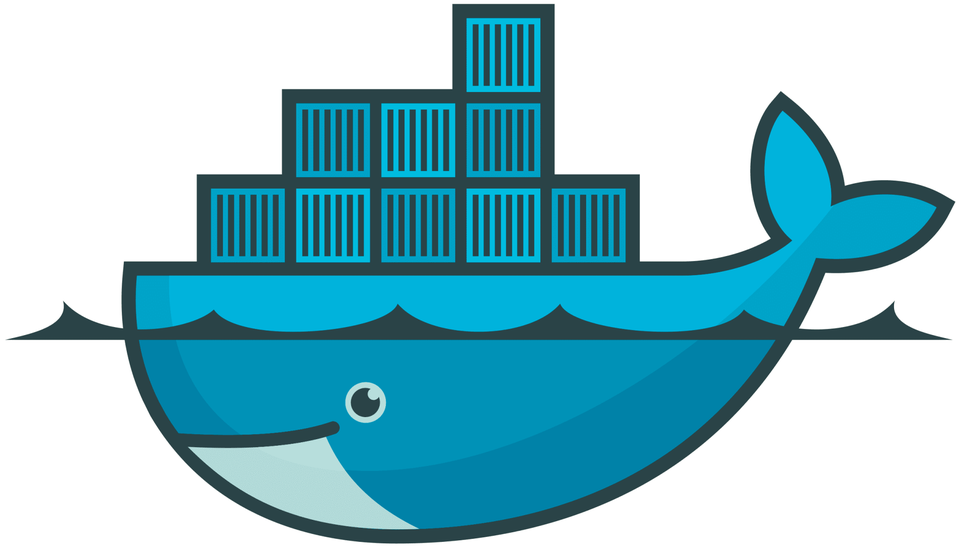
Deployment Examples

- App deployment to WebAPP
- ARM template Deployment

Exercises for Section 4

- Create ARM deployment stage with ADO pipeline
- Create resources using terraform with ADO pipeline
- Include security scanning for Resources using Code
- DependaBot, security Enablement in Github Repos

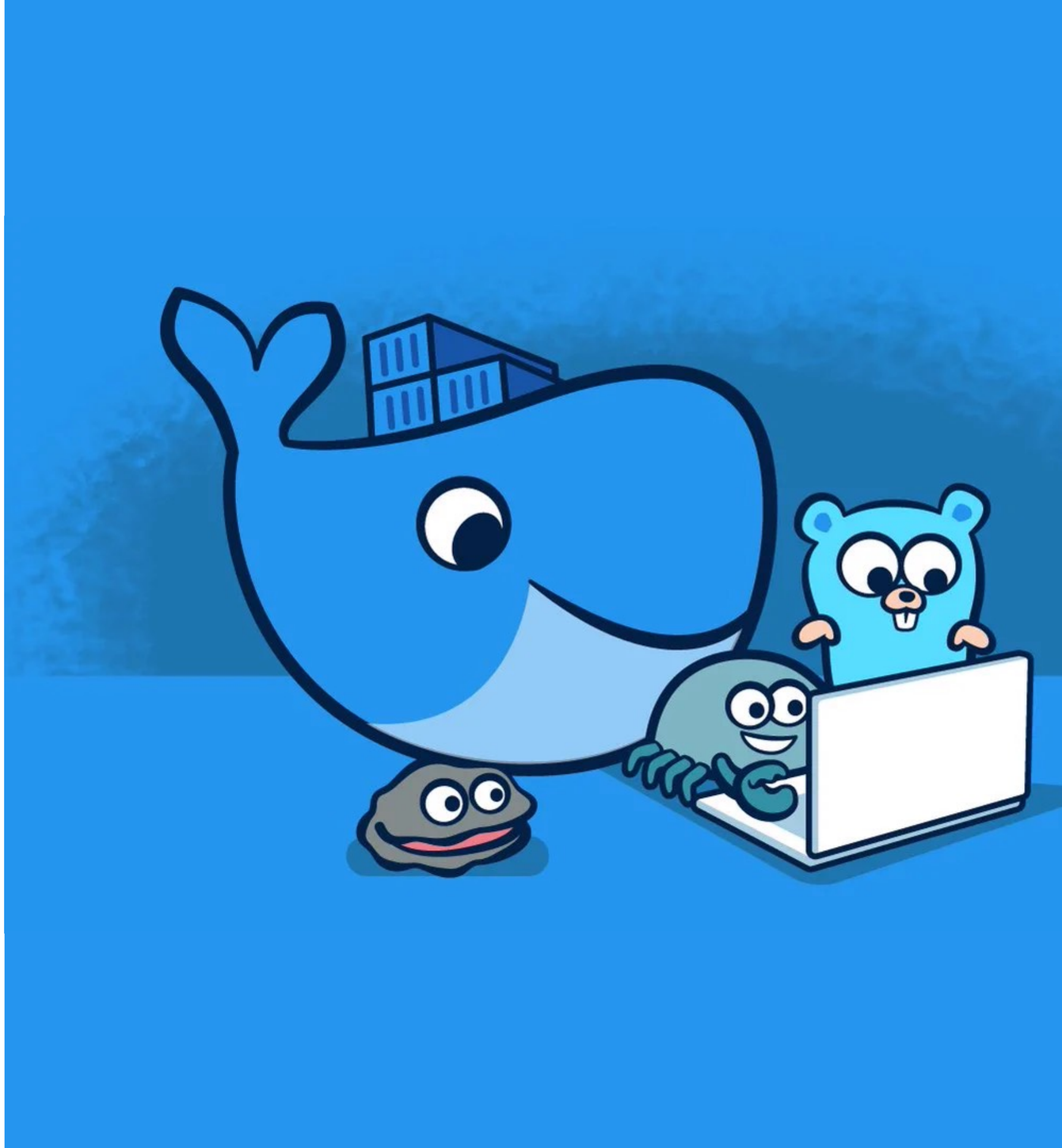
Section 5



docker

Docker setup

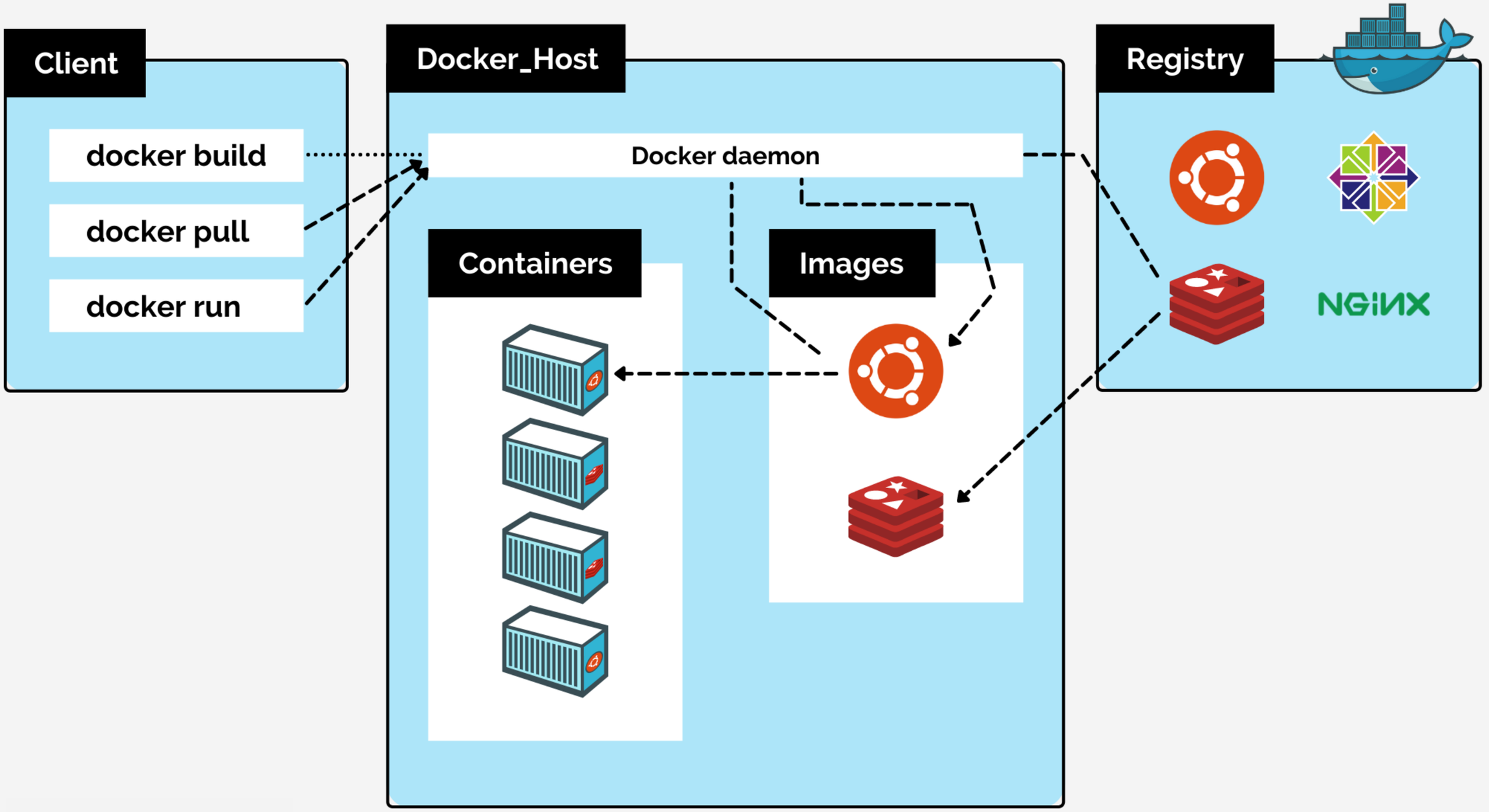
- Docker Desktop in Local machine
- Docker services in Azure Cloud
- ACR, ACI
- PlaywithDocker



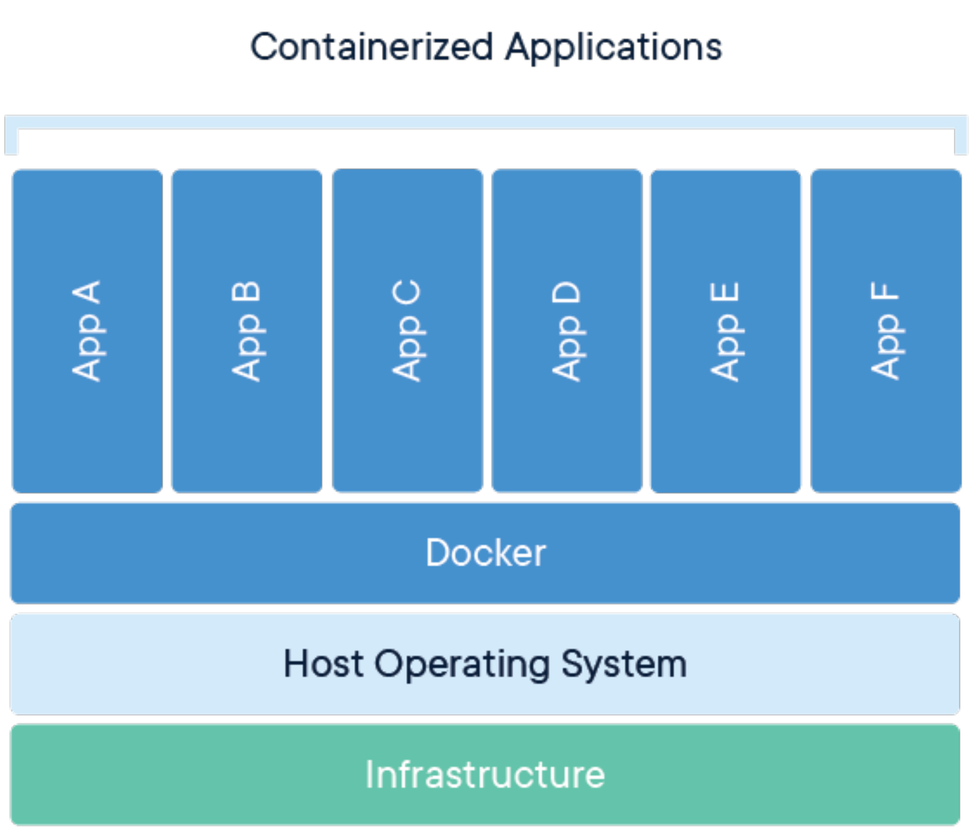
Docker Key Concepts

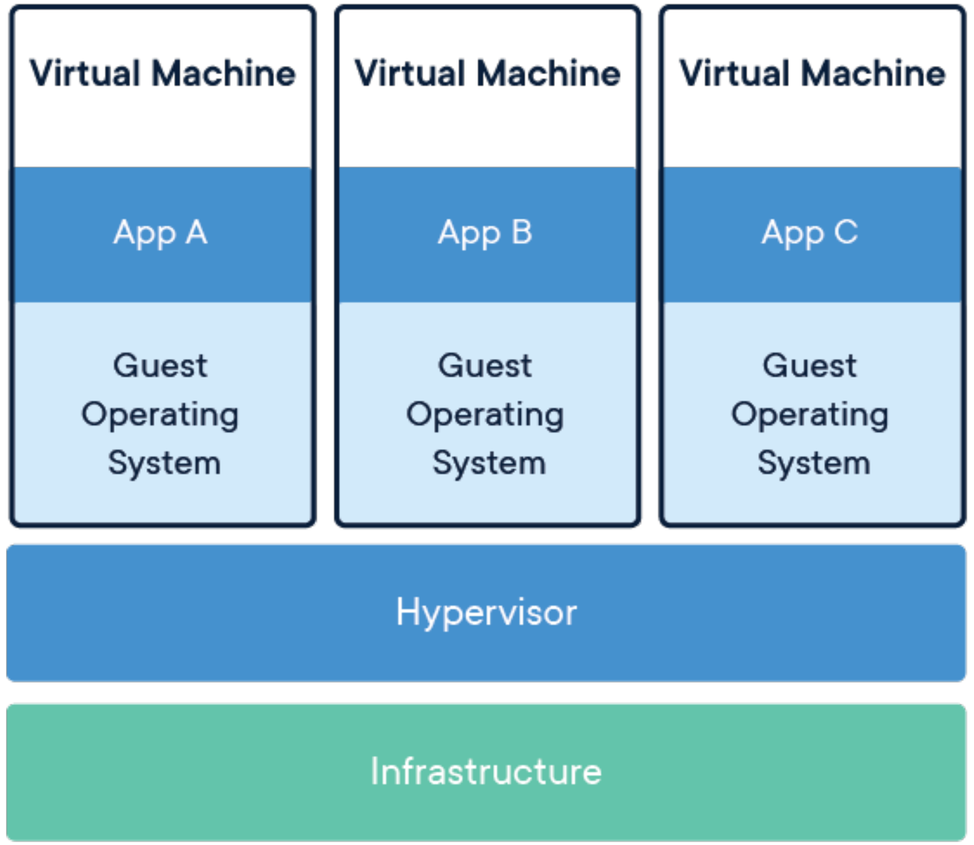
- DockerFile
- Image
- Container
- DockerRegistry
- Docker Filesystem
- Docker Commands
- Docker Compose
- Docker Networking
- Docker Swarm
- Kubernetes

Docker Architecture



Container vs VM architecture

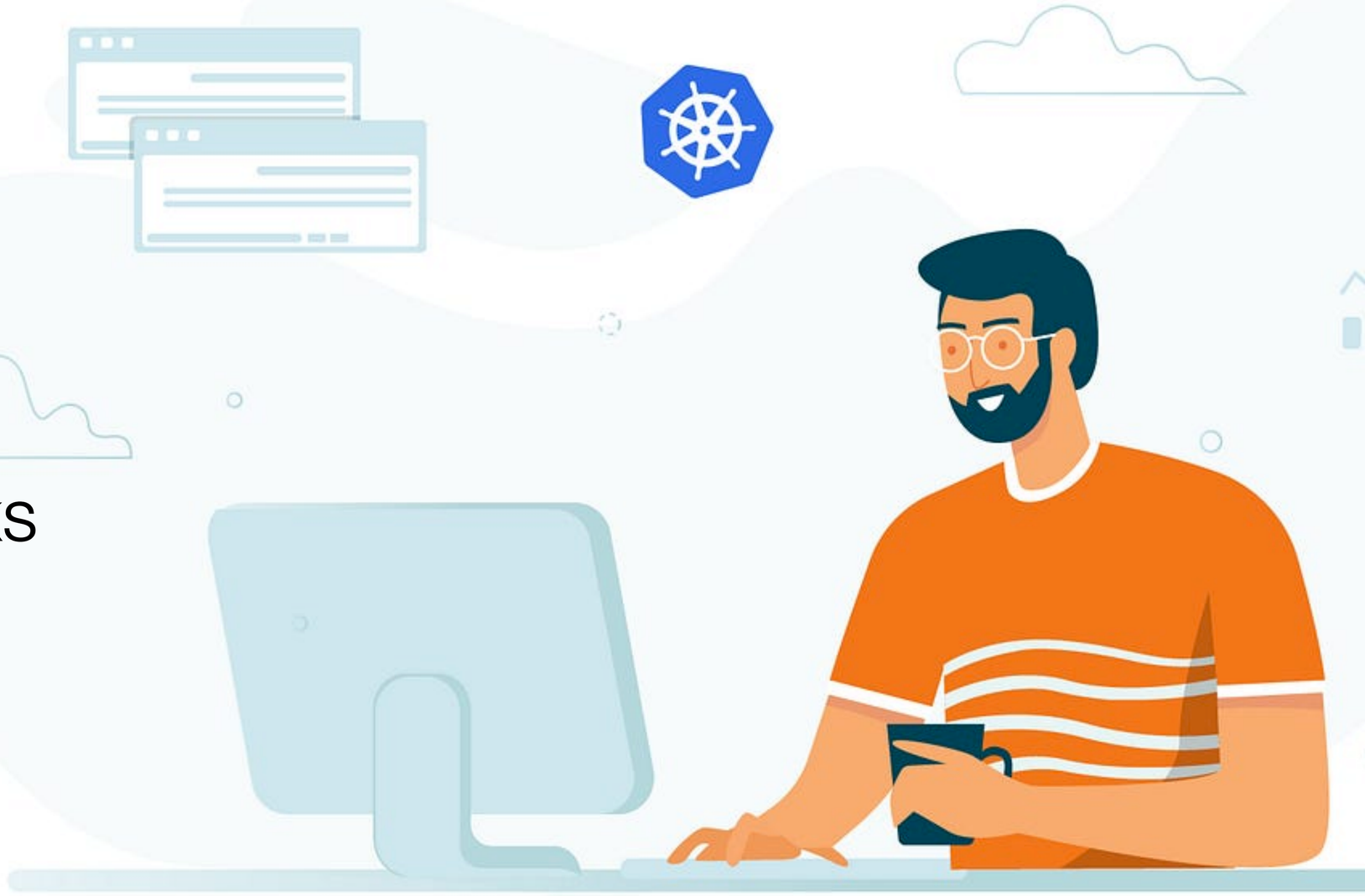




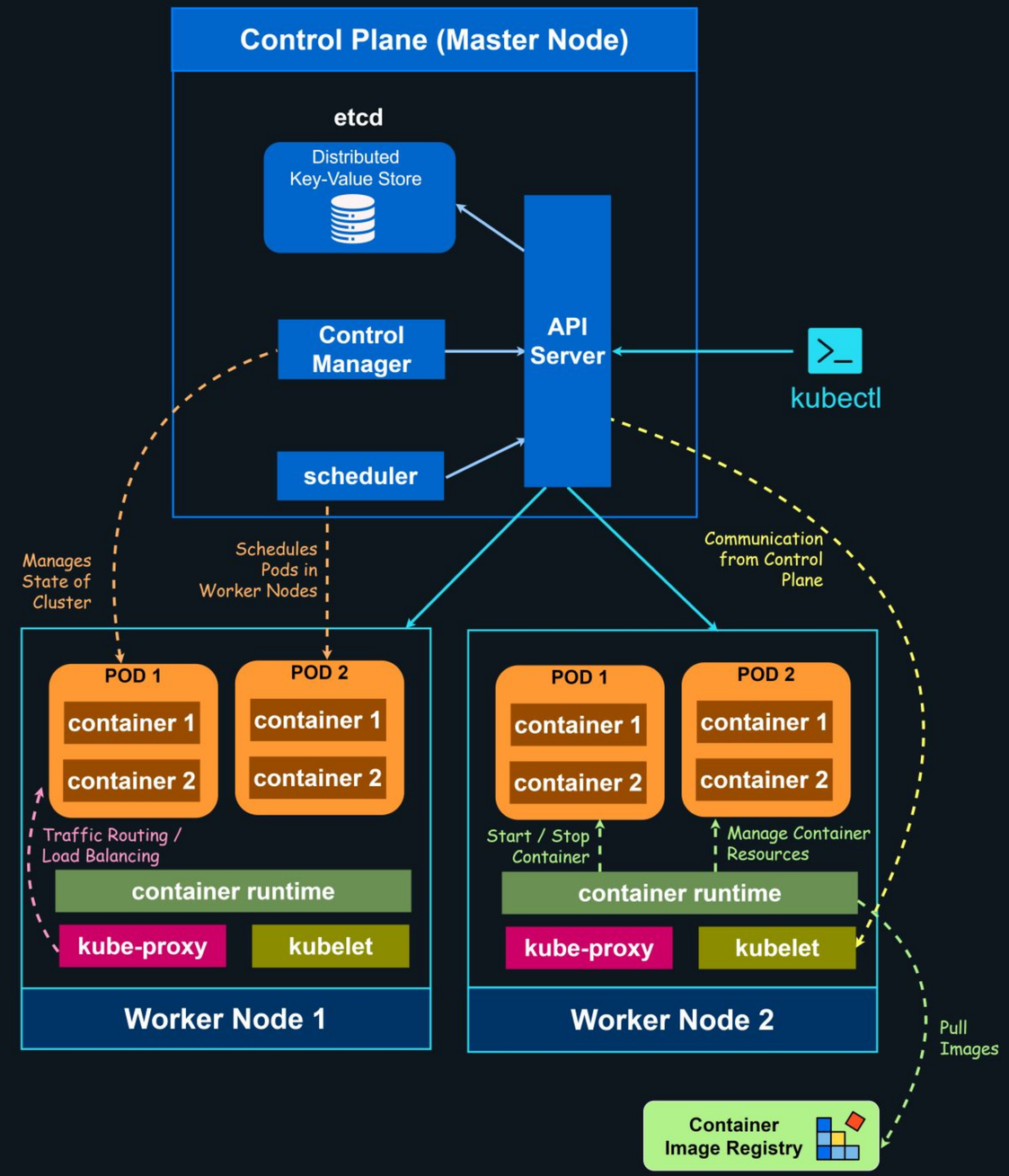
kubernetes

Kubernetes setup

- miniKube
- Docker Desktop
- Kubernetes extension
- Play with k8s
- Azure cloud service - AKS



Kubernetes Architecture Knowledge Map

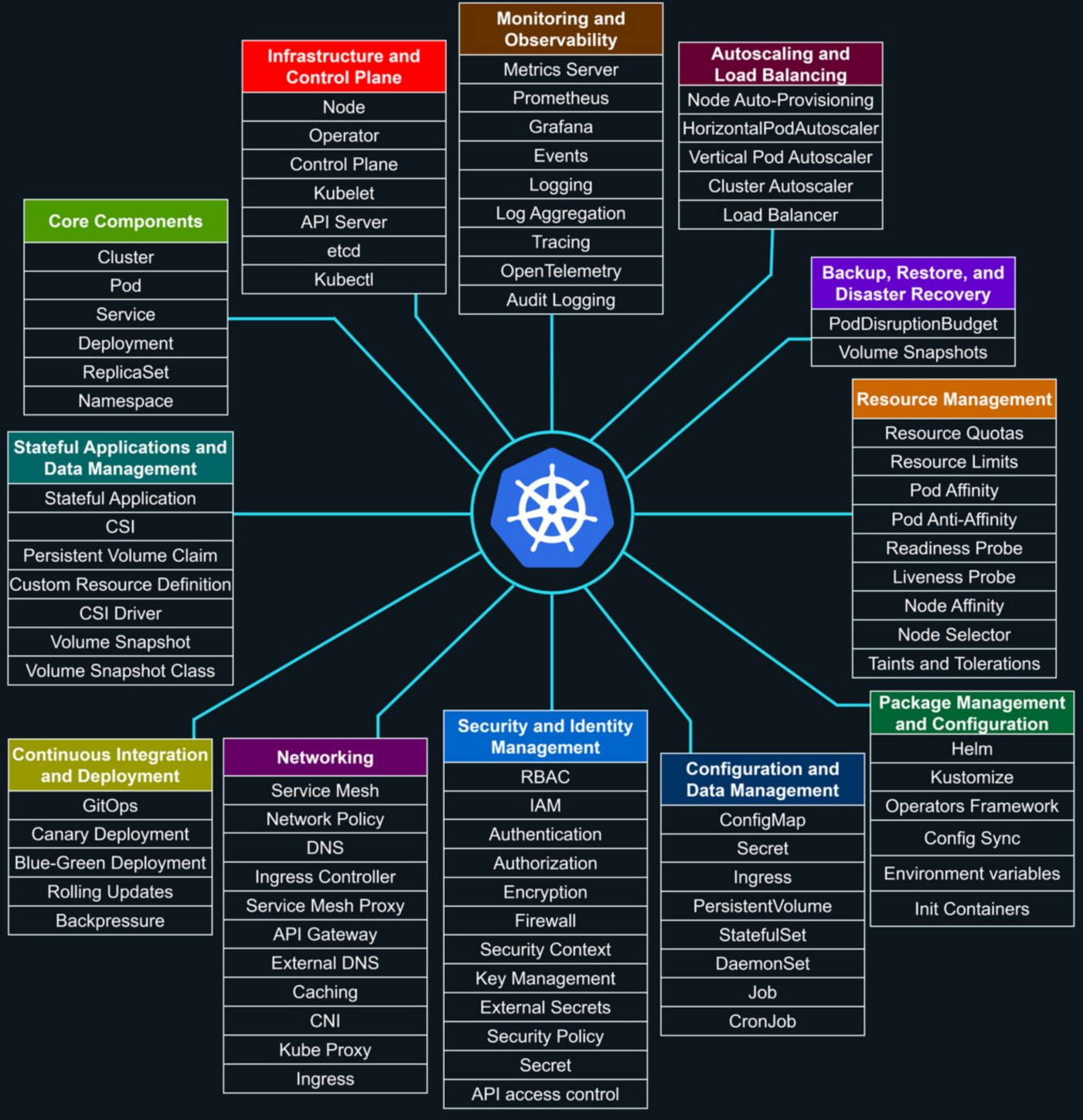


Kubernetes Key Concepts

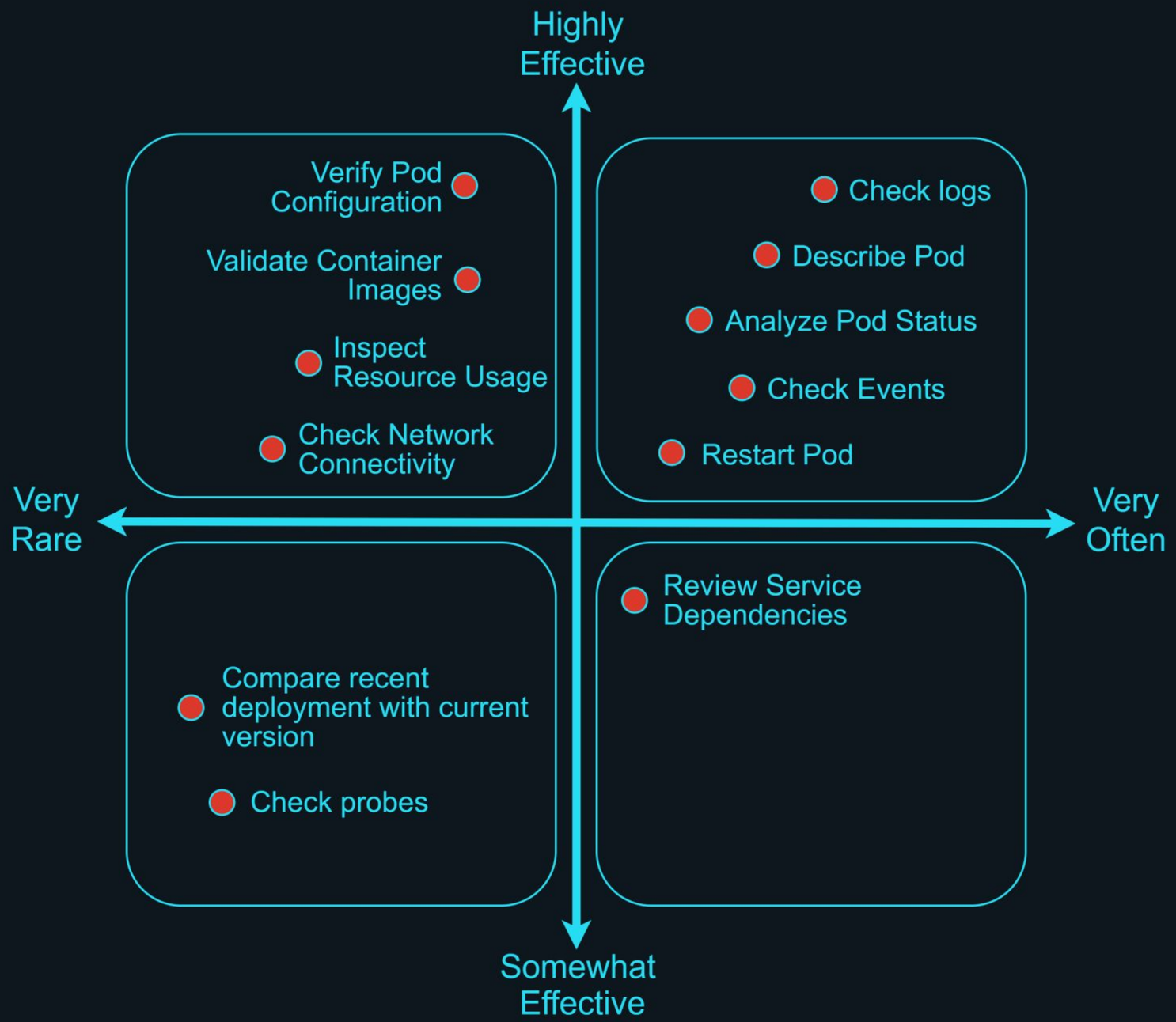
- Pods
- ReplicaSets
- Deployments
- Services
- Imperative way
- Declarative way
- Persistent storage
- Secrets
- Taints
- NodeAffinity
- Cordon
- Ingress

10-STEP KUBERNETES LEARNING PATH

Step	Coverage	Concepts
 **Understand the Basics**	• Concepts and Terminology • Kubernetes Architecture	• Pods, Services, Nodes, Clusters, Deployments, and more • Architecture of Kubernetes, including the control plane and worker nodes
 **Setting Up Your Environment**	• Local Setup • Cloud Setup	• Minikube, Docker Desktop with Kubernetes • EKS, GKE, AKS etc
 **Deploy & Manage Applications**	• Pods • ReplicaSets and Deployments • Services	• Create and Manage Pods • Create ReplicaSets and Deployments to manage scaling and apps updates • Explore how Services expose your apps in-cluster or externally
 **Storage and Configurations**	• Persistent Storage • Configuration Management	• Master PVs and PVCs for app storage • Explore ConfigMaps and Secrets for managing app configuration data
 **Advanced Topics**	• Networking • Security • Monitoring and Logging	• Explore networking, including Ingress, policies, and Service meshes • Study RBAC and security policies • Explore tools like Prometheus, Grafana, Elastisearch etc.,
 **Helm and Operators**	• Helm • Operators	• Learn Helm to package, share, and deploy applications • Understand Kubernetes Operators for automating complex , stateful apps
 **Practice and Projects**	• Hands-On Projects • Troubleshooting	• Create apps on Kubernetes, starting simple and going bigger • Learn how to troubleshoot common issues and problems
 **Community and Resources**	• Join the Community • Documentation and Tutorials	• Participate in Kubernetes forums, mailing lists, and meetups to learn from others • Stay updated with Kubernetes official documentation, blogs, and tutorials
 **Validation**	Certifications	Consider taking the Certified Kubernetes Administrator (CKA) or Certified Kubernetes Application Developer (CKAD) exams
 **Keep Learning**	Stay Informed	Kubernetes is rapidly evolving, so keep learning and experimenting with new features and best practices


Monitoring and Observability
Metrics Server
Prometheus
Grafana
Events
Logging
Log Aggregation
Tracing
OpenTelemetry
Audit Logging
Infrastructure and Control Plane
Node
Operator
Control Plane
Kubelet
API Server
etcd
Kubectl
Autoscaling and Load Balancing
Node Auto-Provisioning
HorizontalPodAutoscaler
Vertical Pod Autoscaler
Cluster Autoscaler
Load Balancer
Core Components
Cluster
Pod
Service
Deployment
ReplicaSet
Namespace
Backup, Restore, and Disaster Recovery
PodDisruptionBudget
Volume Snapshots
Stateful Applications and Data Management
Stateful Application
CSI
Persistent Volume Claim
Custom Resource Definition
CSI Driver
Volume Snapshot
Volume Snapshot Class
Resource Management
Resource Quotas
Resource Limits
Pod Affinity
Pod Anti-Affinity
Readiness Probe
Liveness Probe
Node Affinity
Node Selector
Taints and Tolerations
Continuous Integration and Deployment
GitOps
Canary Deployment
Blue-Green Deployment
Rolling Updates
Backpressure
Networking
Service Mesh
Network Policy
DNS
Ingress Controller
Service Mesh Proxy
API Gateway
External DNS
Caching
CNI
Kube Proxy
Ingress
Security and Identity Management
RBAC
IAM
Authentication
Authorization
Encryption
Firewall
Security Context
Key Management
External Secrets
Security Policy
Secret
API access control
Configuration and Data Management
ConfigMap
Secret
Ingress
PersistentVolume
StatefulSet
DaemonSet
Job
CronJob
Package Management and Configuration
Helm
Kustomize
Operators Framework
Config Sync
Environment variables
Init Containers

Kubernetes POD Troubleshooting Techniques



ChatGPT

Exercises for Section 5

- Installation of Docker, Minikube , Docker Kubernetes extension
- Create docker image using DockerFile
- Store container data in volumes
- Deploy kubernetes yaml files in minikube and AKS
- Expose application to access outside
- Use ChatGPT to generate snippets for various components - Azure Devops, Docker, Kubernetes