Chmurowisko

www.chmurowisko.pl

OpenShift

AGENDA

Chmurowisko

1. What is OpenShift?
2. First login
3. Quick start with OpenShift
4. Application Build
5. Deployment
6. OpenShift Architecture

ASSUMPTIONS

Chmurowisko

1. We all work on a single instance of Azure Red Hat OpenShift (ARO)
2. We will explore all functions from the Dev/DevOps point of view
3. We will concentrate on a practice
4. Ask as soon as you want – all of us are here to gain valuable knowledge

REQUIREMENTS

- Azure AD account (*@chmurowiskolab.onmicrosoft.com*)
- Github user account (https://github.com/).

  If you don't have an account follow these steps:

  1. Go to https://github.com/ and click on "Sign up"
  2. Choose any username. It can be your student login e.g.: studentXX (where XX is your number)
  3. Provide real e-mail address (it'll be used to confirm registration)
  4. Provide a password (8 signs, letters & numbers)
  5. Fill short Github's survey (if it'll be provided to you)
  6. Check your email and confirm registration

EXERCISE

Chmurowisko

Did you use OpenShift previously?

What features did you use?

What is **OpenShift?**

1 – What is OpenShift?

Definition

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

1 – What is OpenShift?

Definition

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

1 – What is OpenShift?

Definition

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

1 – What is OpenShift?

Definition

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

1 – What is OpenShift?

Why it is based on Docker?

What are the pros of Docker (from developer point of view)?

- Easy application shipment
    - Applications are packed into Docker images (isolated environments)
    - Applications are ready to be run everywhere (local machine, on-prem, cloud, staging environment, production)

1 – What is OpenShift?

Why it is based on Docker?

Pros of easy application shipment:

- MTTM (Mean Time To Market) is lower (*lower = better*)
- Easier continuous integration & deployment (CI/CD)
- Container runs the same on every environment
- Applications tend to be smaller and designed to be independent of each other

1 – What is OpenShift?

Why is it based on Kubernetes?

- What is Kubernetes?
  - Framework for running containers
  - Foundation for effective workload management
- Kubernetes pros:
  - ...

1 – What is OpenShift?

Why is it based on Kubernetes?

- Kubernetes pros:
  - Application scalability and failover
  - Workload deployment standard
  - Service discovery
  - Load balancing
  - Storage orchestration & management
  - Rollouts
  - Resource quotas
  - Rescheduling, performance optimalizations
  - Self-healing
  - Secret & Configuration management

1 – What is OpenShift?

Whi it is based on Docker and Kubernetes?

- These tools creates a standard:
  - Deployment standard
  - Running and managing lifecycle of a container

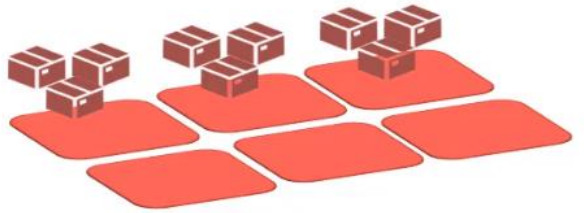
1 – What is OpenShift?

Why it is based on Docker and Kubernetes?

OPENSHIFT Tools

Kubernetes

Docker



1 – What is OpenShift?

Definicja

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

1 – What is OpenShift?

Definicja

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

1 – What is OpenShift?

Why OpenShift exist?

- Running containers on clusters in production environments requires additional knowledge, tools and resources (hardware, people)
  - Container images registry
  - CI/CD pipelines
  - Logging & monitoring tools (Prometheus, Grafana)
  - Storage management
  - Networking
- We need to configure, tests and maintenence every additional tool we install on a cluster
- OpenShift extends Kubernetes and packing all useful tools into single platform

1 – What is OpenShift?

Who are the users of OpenShift?

- Whole organizations, not single teams
- OpenShift concentrate on helping:
  - Developers (Dev)
  - Administrators (Ops)

1 – What is OpenShift?

Who are the users of OpenShift? - Developers

- (Generally speaking) Technical tasks solved by developers are:
  - Creating the application
  - Implementing new features
  - Testing features
  - Deploying applications
- In other words: Developers job is to deliver working features to end users as soon as possible

1 – What is OpenShift?

Benefits of OpenShift for developers

- OpenShift give developers:
  - Tools for building Docker images from code (even without Dockerfiles)
  - Built-in container image registry
  - Automatization of build and deployment process
  - Standarized way of deploying applications (across whole organization)
  - Tools for collecting logs and monitoring workloads
  - Platform for running workloads effortlessly
- OpenShift give developers tools required to deliver new features faster, sooner and with less pain (coming from integrating additional 3rd party tools)

1 – What is OpenShift?

Who are the users of OpenShift? - Administrators

- Major tasks
  - Maintaining, looking after applications
  - Keeping the application highly available (HA)
  - Scaling the application
  - Taking care of infrastructure

1 – What is OpenShift?

Benefits of OpenShift for administrators

- What are the benefits?
  - Single, standarized way of creating, deploying, running and managing applications across different teams
  - Better isolation of applications than using plain Kubernetes (thanks to Project abstraction)

1 – What is OpenShift?

Definicja

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

FIRST LOGIN TO OPENSHIFT

2 – First login to OpenShift

Cluster creation

- We will use Microsoft Azure and Azure Red Hat OpenShift (ARO) service
  - ARO: https://azure.microsoft.com/en-us/services/openshift/
  - OpenShift managed by Azure
  - Integration with other Azure services
  - SLA – 99.95%
- Requirements
  - 34 vCore per installation (6 VM; 3x D8s v3 + 3x D4s v3)
  - Cost: 1635$/mo (Pay As You Go)

2 – First login to OpenShift

Requirements to install cluster on Azure

- User need following roles:
  - On subscription level: at least Contributor (the best: Owner) + User Access Administrator role
  - On AAD level: Application Administrator or Global Administrator
- Currently ARO service can be created only from `az cli`
- Installation takes up to 35 minutes
- Similarly to Kubernetes, there will be 2 resource groups created for you

2 – First login to OpenShift

Demo: Login as cluster administrator

- Get the cluster URL:
  - az aro show –r <resource_group_name> -n <cluster_name>
- After you create ARO you need to get Cluster Admin credentials:
  - az aro list-credentials –r <resource_group_name> -n <cluster_name>

2 – First login to OpenShift

Integrate OpenShift with Azure AD

- Inside AzureActive Directory you should:
  - Add new App Registration
  - Add new Client Secret
  - Add Optional Claims
- In OpenShift you should:
  - Configure Identity Provider for Azure

2 – First login to OpenShift

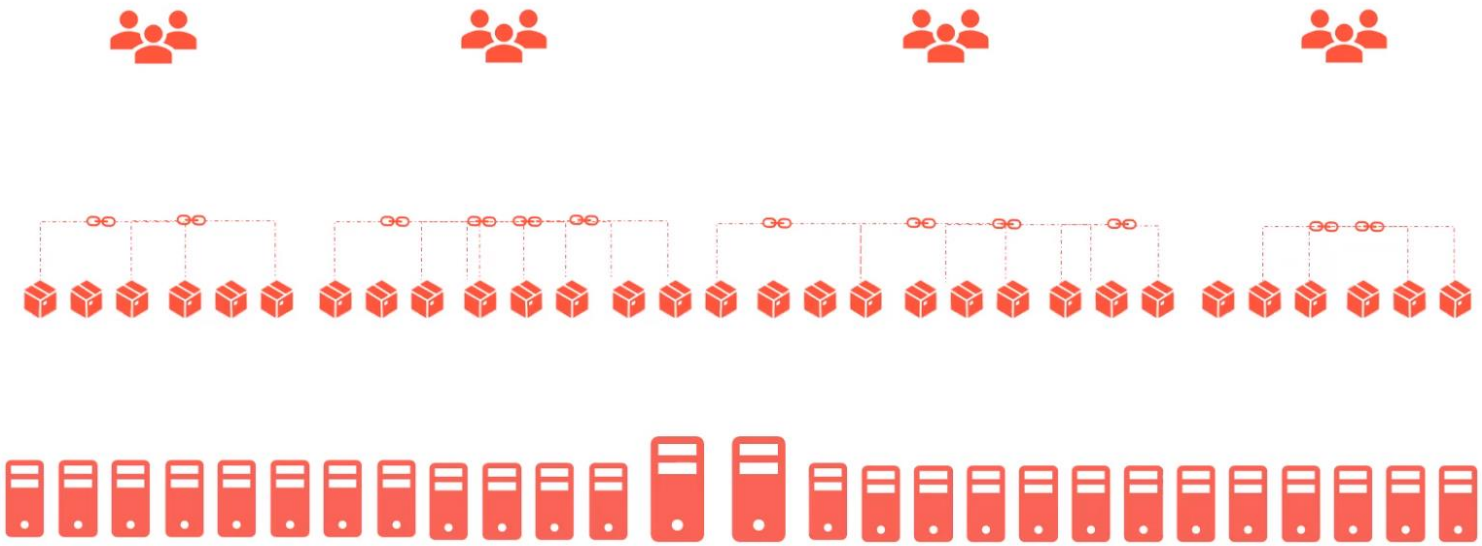
UI demonstration

- Perspectives
  - Developer
  - Administrator
- Perspective is a way to narrow down Web Console UI to most important options from single role point of view
- Perspective is not a role in a cluster

2 – First login to OpenShift

Projects

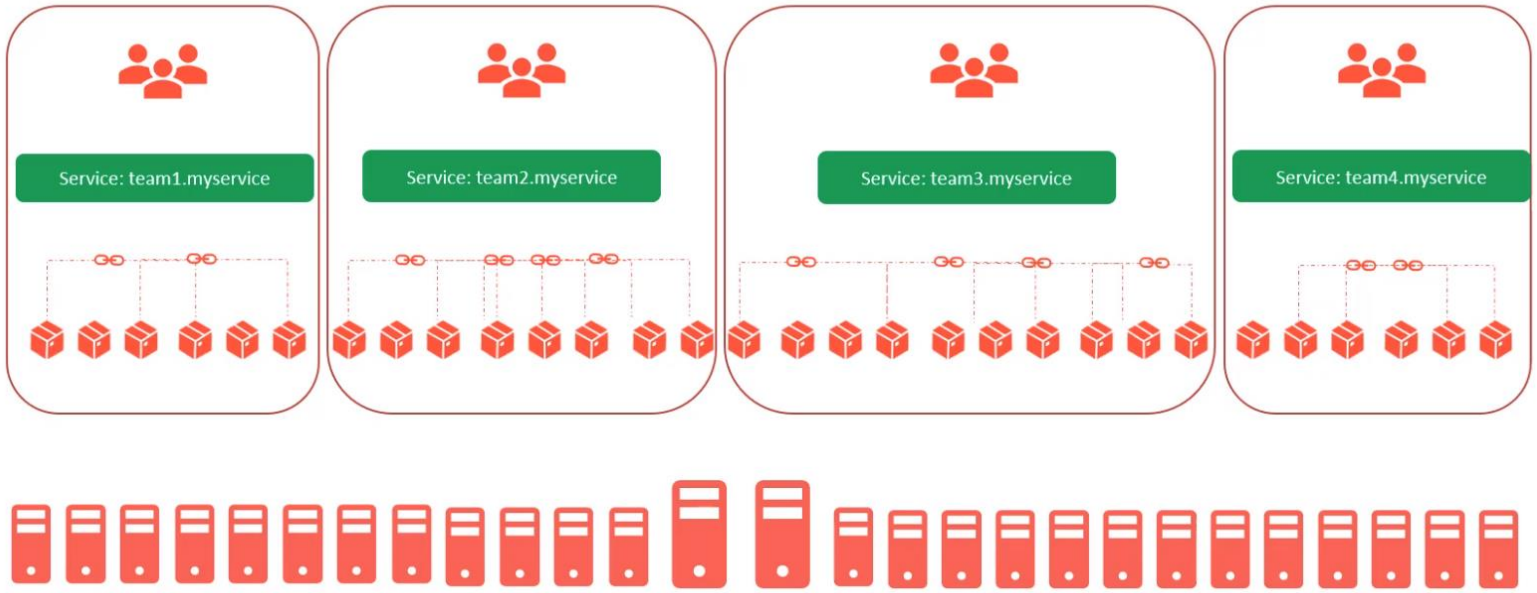
- Projects are used to separate teams and their applications from each other



2 – First login to OpenShift

Projects

- OpenShift introduce Project abstraction over Kubernetes namespaces
- Project is isolated space for a single team or a single application in your portfolio



2 – First login to OpenShift

Users basics

- There are 3 types of users in OpenShift:
  - Regular
  - System
  - Service Accounts

2 – First login to OpenShift

Authentication

- OpenShift has it own OpenID Connect server that handles authentication and authoriziation of users
- Azure AD acts as 3rd party Identity Provider (orhogonal to OpenShift). It provides the login screen, verifies the user against its own database and then passes the user identity to OpenShift in form of JWT token. Then OpenShift matches user from AAD with user instance inside its own database. Then OpenShift uses the user instance it gets from own database to authorize each action.
- All actions in OpenShift are authorized

2 – First login to OpenShift

First login

- You should login to OpenShift using account in *@chmurowiskolab.onmicrosoft.com* accounts
- Link to login I'll send on a chat

QUICK START

3 – Quick start

Lab

- Lab: 10-getting-started
- Time: 20-30 minutes

3 – Quick start

Exercise - summary

- What we achived?
  - OpenShift automatically pulled code from remote repository
  - OpenShift created new Container Image
  - OpenShift shared the Image in internal Docker Image Registry
  - OpenShift automatically deployed application

3 – Quick start

OpenShift from cluster administator point of view

- Regular users has no access to each other projects
- Cluster Administrator has access to all projects
- Cluster Administrator can impersonate user
    - In case of any problems I'll use this feature to help you with further exercises

3 – Quick start

Adding teammates to projects

- Project creator, Cluster Administrator or any person with enough permissions inside a project can add teamates to it

3 – Quick start

Demo – using OpenShift through CLI

- Log into OpenShift through CLI
- Requirements
  - `oc` tool installed (https://docs.openshift.com/container-platform/4.6/cli_reference/openshift_cli/getting-started-cli.html)
- Example commands
  - oc get projects
  - oc project <project name>
  - oc get dc
  - oc rollback

APPLICATION BUILD

4 – Application Build

What is Application Build?

- Application Build is a process of transforming application code into the Docker image.
- Application Build process is described as BuildConfig object
- Each Application Build has:
  - It accepts a defined input (e.g.: build trigger)
  - It logs its standard output (stdout)
  - It publishes created image in ImageStream (if build succeed)
  - It presents its own build status
- Build can be subject of resource limiting

4 – Application Build

BuildConfig – what is it?

- Declerative way of describing how to build container image
  - Feature that is missing in Kubernetes
- Object in OpenShift
  - It knows how to receive webhook from SCM
  - It knows how to build container image
  - It knows how to place container image in Container Image Registry
    - Internal registry
    - When configured it can push images to other registries

4 – Application Build

3 strategies of building images

- Basic
  - Source-To-Image Build (S2I)
  - Docker Build
- Advanced
  - Custom build
    - Used to build specialized artifacts (e.g.: base images)

4 – Application Build

Source-To-Image (S2I)

- Simplest way of running an application on OpenShift
- You don't have to provide a Dockerfile
    - OpenShift doesn't need application to be contenerized – it can contenerize it on its own

4 – Application Build

Why you would use Source-To-Image?

- To use run non-contenerized application on OpenShift
- Speed
  - S2I is faster than Dokcer build (thanks to OpenShift optimalizations)
- Security
  - S2I applies additional security options to make your images more secure (e.g.: make container use non-root user)

4 – Application Build

Docker Build

- The simplest build strategy
- The code repository need to contain Dockerfile

4 – Application Build

Lab: Docker Build

- Lab: "11-deploy-dockerfile-application"

4 – Application Build

Build Trigger

- 3 types of build trigger
  - Webhook
  - ConfigChange
  - ImageChange

4 – Application Build

Lab: Build Trigger

- Lab: 12-add-build-trigger

4 – Application Build

Getting code from private Git repository

- https://www.openshift.com/blog/deploy-private-git-repositories
- Repository must be reachable for OpenShift cluster

4 – Application Build

BuildConfig – pros

- Source-To-Image knows how to build container images for your application (even if it has no Dockerfile)
- It knows how to get code from private repositories
- Application build is run inside OpenShift – code doesn't go outside your environment

4 – Application Build

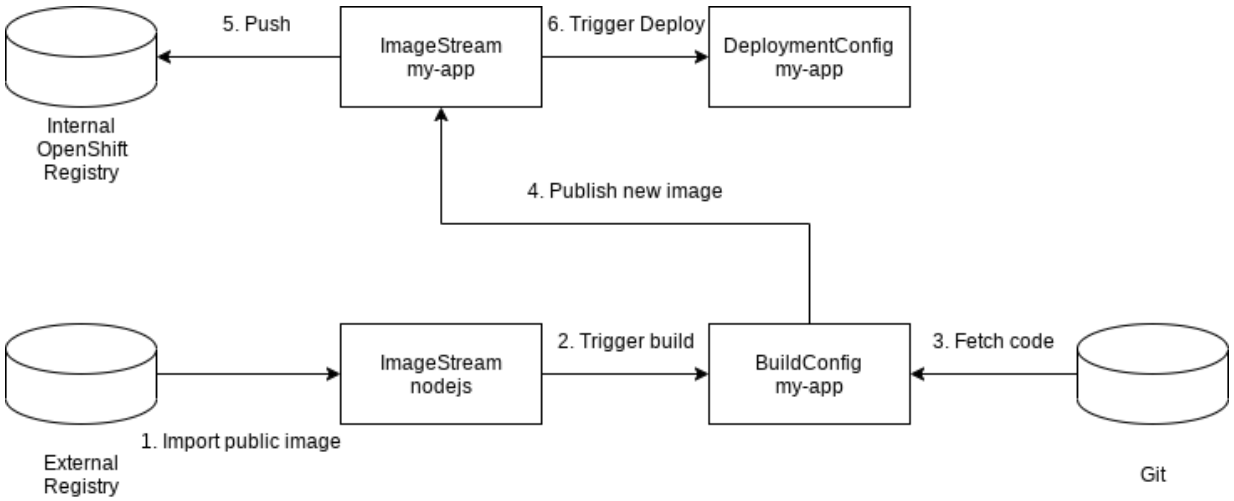
ImageStream – what is it?

- OpenShift feature for storing and versioning container images
- It maps tags to concrete image instance
- It is used as a source of images for Deployments
- It can import images from other registries (e.g.: Docker Hub) and store them inside of itself

4 – Application Build

ImageStream - pros

- OpenShift updates ImageStream automatically after each Build (switching Docker image under selected tags), therefore triggering Deployment.
- Abstraction of managing container images registry
- Useful scenario: base image update triggers build of every dependent image

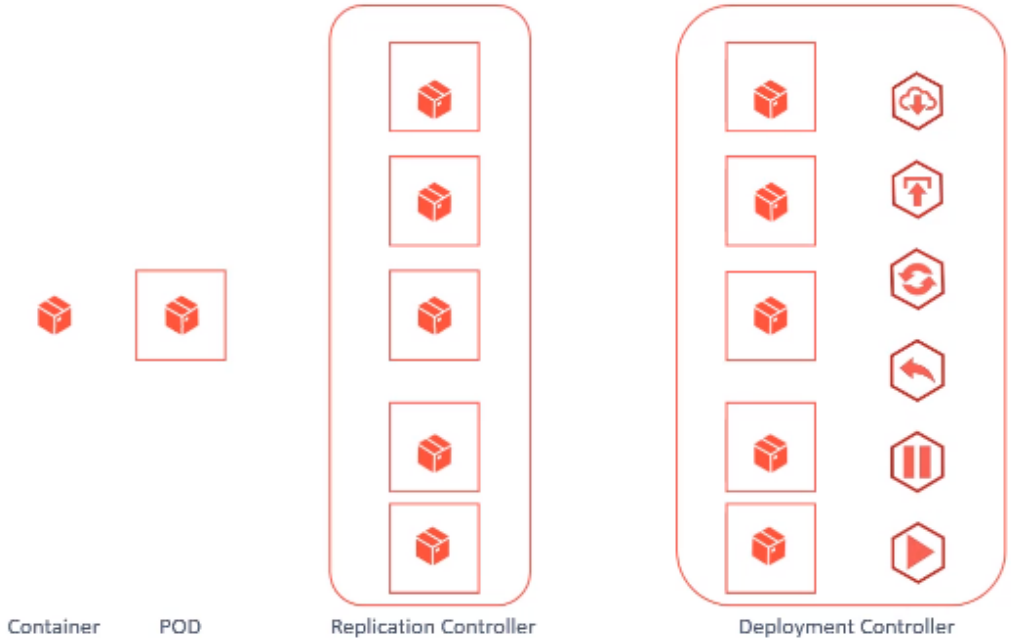


DEPLOYMENT

5 – Deployment

What is Deployment?

- Pod is the smalles and simples unit you can deploy
- OpenShift uses Replication Controller (instead of Kubernetes ReplicaSets)
- Deployment
  - It is Kubernetes Deployment
  - Uses container images from Image Stream instead of Docker Hub

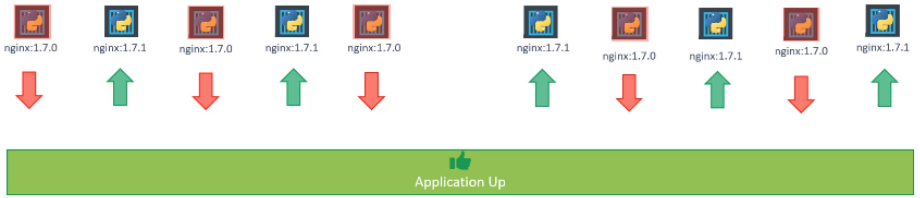


5 – Deployment

Deployment Strategies

- Recreate

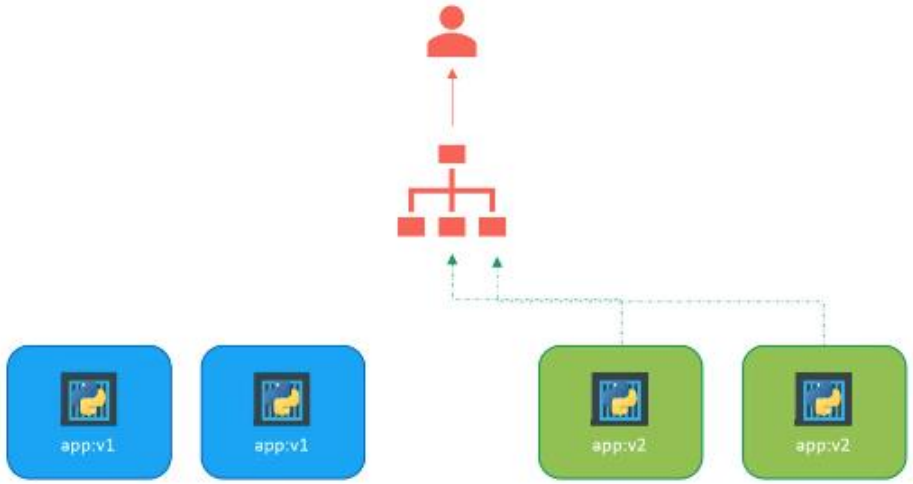
- Rolling (default)



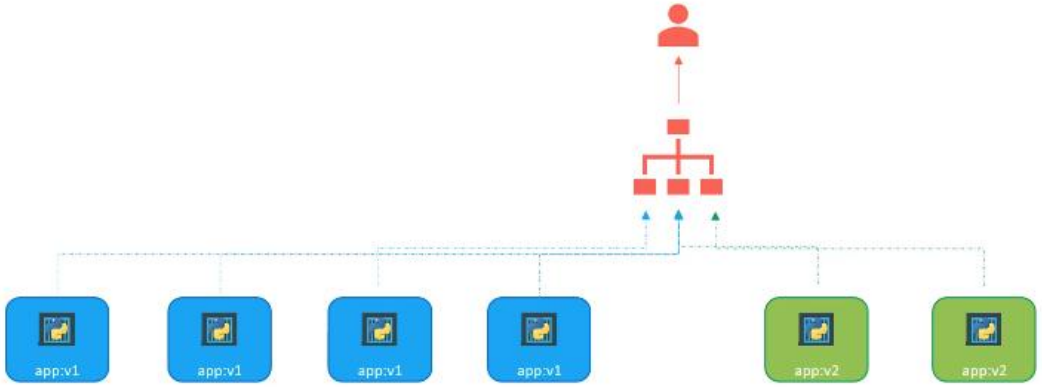
5 – Deployment

Advanced Deployment Strategies

- Blue/Green



- A/B



5 – Deployment

Difference between Deployment and Deployment Config

- Deployment
  - Kubernetes object
- Deployment Config
  - Previous abstraction OpenShift uses to manage Deployments

OPENSHIFT ARCHITECTURE

5 – OpenShift Architecture

Definition

OpenShift is a platform for contenerized applications.

It is developed by Red Hat.

It uses Docker and Kubernetes under the hood.

It was created for enterprise grade systems and applications.

5 – OpenShift Architecture

OpenShift versions

1. **Azure RedHat OpenShift**
2. OpenShift Origin - open source (https://www.okd.io/)
3. OpenShift Online
4. OpenShift Dedicated
5. OpenShift Enterprise

5 – OpenShift Architecture

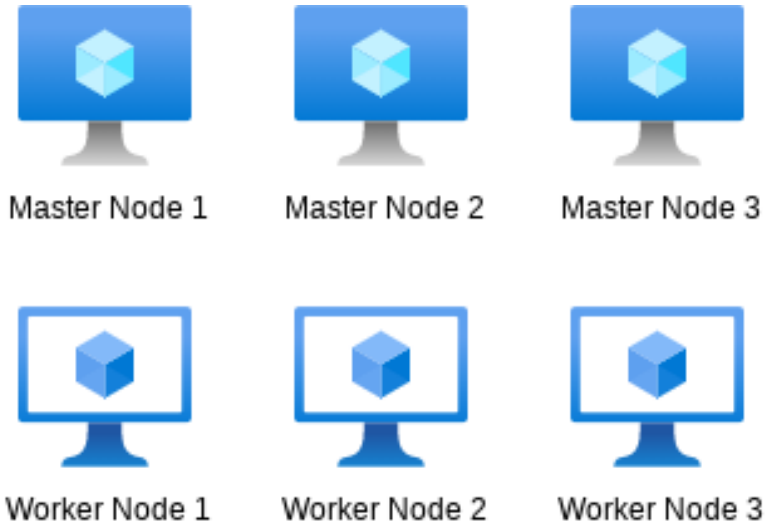
Different installation scenarios

- Setup
  - All-in-One (minishift; development)
  - Single Master/Multiple Workers
  - Mulitple Masters/Muliple Workers
- Different environments
  - On-Premise
  - Cloud
- Various installation scenarios
  - Manually
  - Through a script
  - Managed by provider (ARO)

5 – OpenShift Architecture

OpenShift cluster architecture

- By default ARO cluster has 3 Master Nodes (Control Plane) and 3 Worker Nodes
- It is easy to add new Worker Nodes



5 – OpenShift Architecture

OpenShift cluster architecture

- By default ARO cluster has 3 Master Nodes (Control Plane) and 3 Worker Nodes
- It is easy to add new Worker Nodes

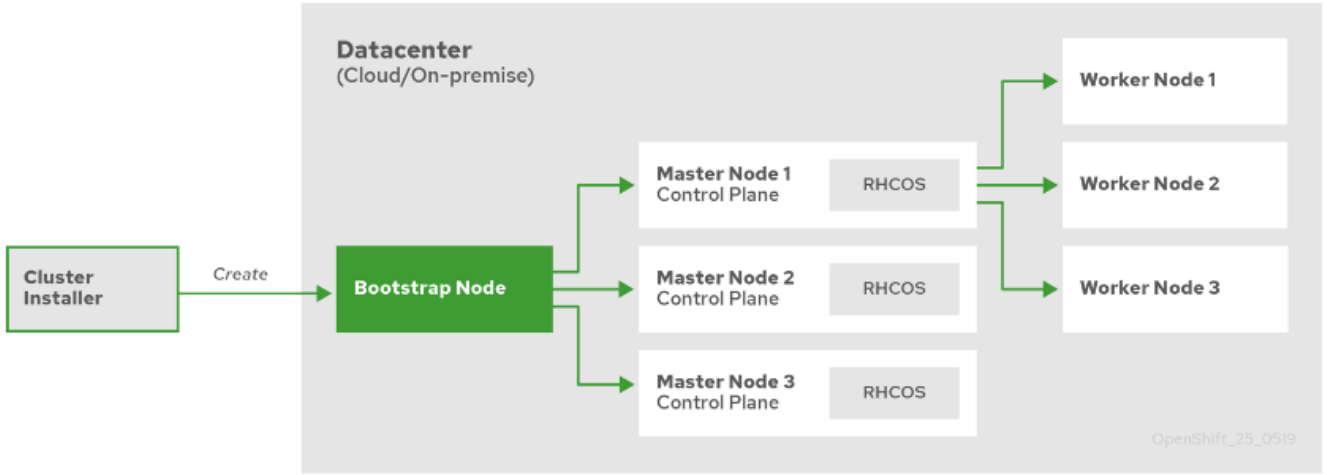
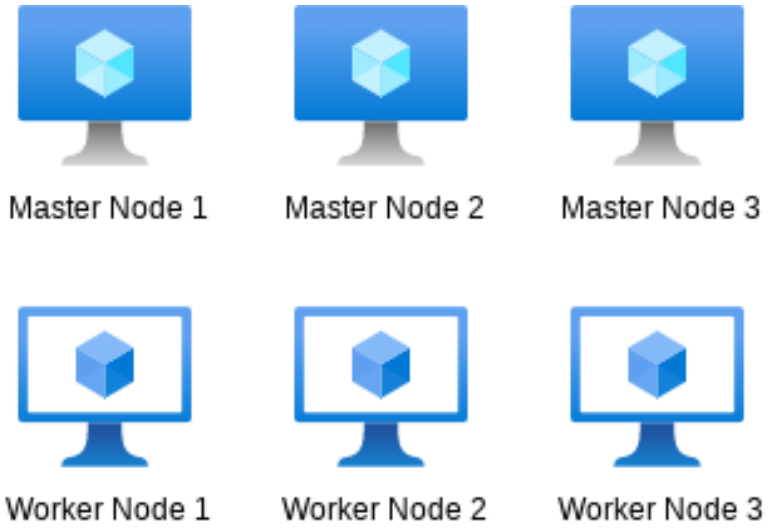


Figure 2. Creating the bootstrap, master, and worker machines



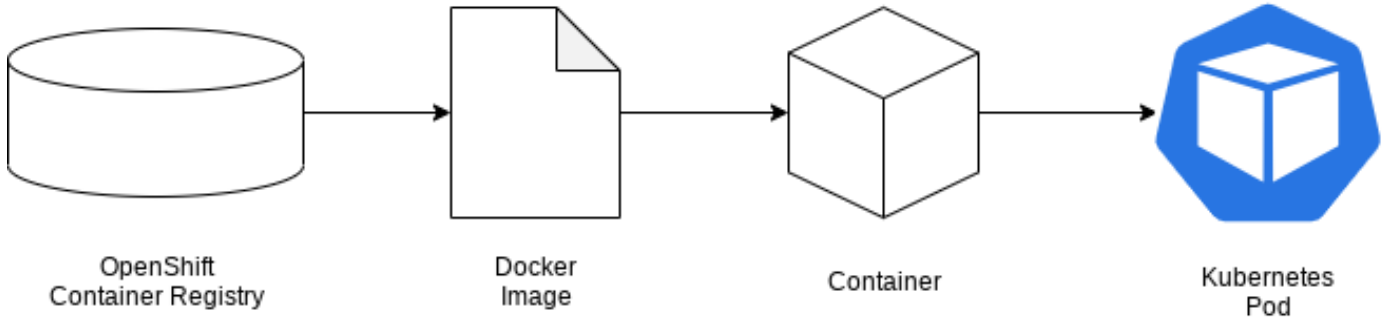
5 – OpenShift Architecture

Demo

Demo: how to add new Worker Node

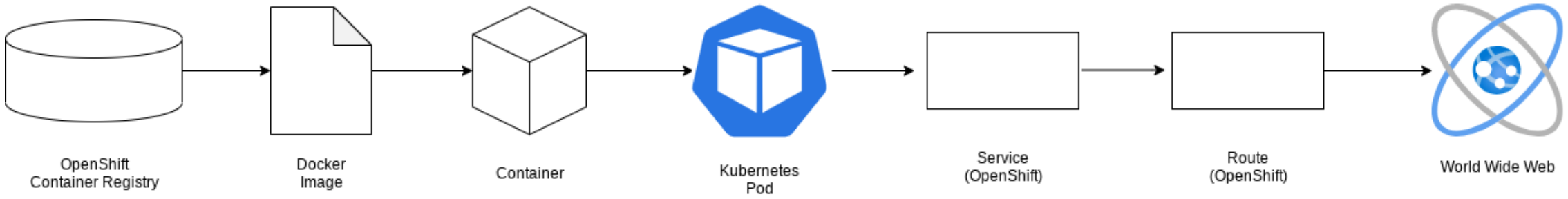
5 – OpenShift Architecture

OpenShift – Docker – Kubernetes relation



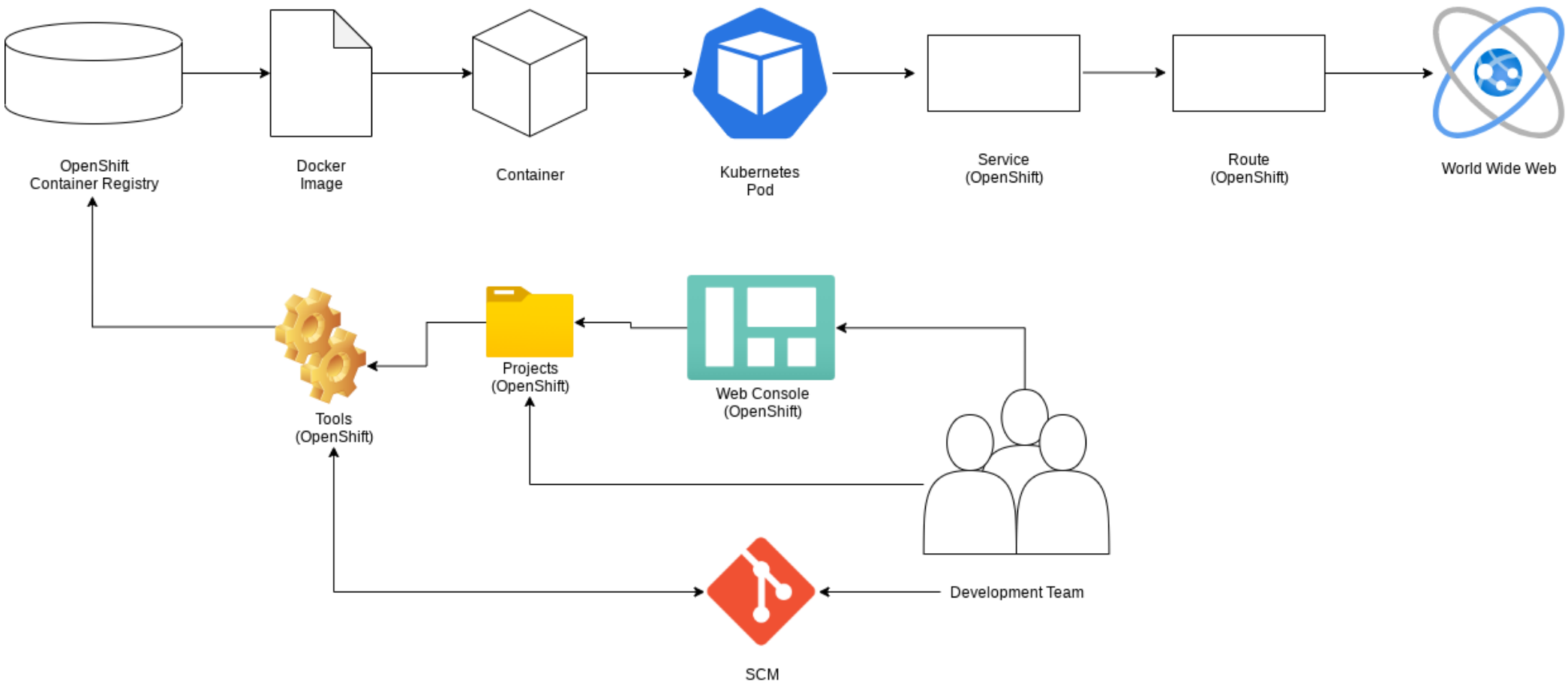
5 – OpenShift Architecture

OpenShift – Docker – Kubernetes relation



5 – OpenShift Architecture

OpenShift – Docker – Kubernetes relation



5 – OpenShift Architecture

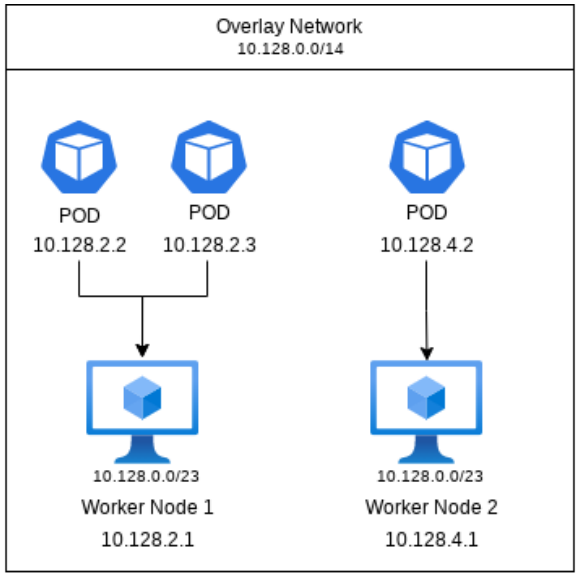
OpenShift – Docker – Kubernetes relation

SERVICES & ROUTES

6 – SERVICES & ROUTES

OpenShift Networking

- Pods need a way to communicate with each other to complete business operations
  - Each Pod in a cluster need unique IP address to communicate with it
- OpenShift use Software Defined Network (SDN) to create and manage network
  - Default: 10.128.0.0/14
  - Each node (master and worker) create unique subnetwork (10.128.0.0/23), so every Pod in thus subnet has unique address



6 – SERVICES & ROUTES

How to connect Pods between each other?

1. IP
2. DNS
3. Service

6 – SERVICES & ROUTES

Service – what is it?

- Service in OpenShift are the same as Services in Kubernetes
- If you want to communicate between two business units (services) you need to wrap them in a Service and this way expose to other services
- If you want to connect to some external service you also do this by Service

6 – SERVICES & ROUTES

Service - benefits

- You can modify Pods inside the Service without worry about destorying underlaying infrstructure
- Each Service gets it's own IP address and DNS name. IP address is of type Cluster IP.

6 – SERVICES & ROUTES

How to connect a Pod into Service?

1. In OpenShift we use Labels and Selectors. Labels are pinned to Pods and Services will select Pods with matching label.
2. You need to provide a port

6 – SERVICES & ROUTES

Service exposes application to cluster internally

- Service can be the an API to internal service or queue, meaning we don't want to expose it to external world
- This is a service premise in OpenShift – it is meant to use for internal communication only

6 – SERVICES & ROUTES

Lab: 21-connect-multiple-services

- https://github.com/cloudstateu/cb-042021

6 – SERVICES & ROUTES

Route

- How to exposes frontend application to external world?
- Route expose internal Service to external world through some known address
- Addtional feature
  - Load balancing
  - Security
  - Traffic splitting

6 – SERVICES & ROUTES

Load Balancing

- 3 Load Balancing strategies
  - https://docs.openshift.com/container-platform/4.6/networking/routes/route-configuration.html#nw-route-specific-annotations_route-configuration
  - source
    - Sticky Session mechanics
    - Requests from single client IP are always routed to same Pod
  - roundrobin
    - Each request is always routed to Pods randomly
  - leastconn
    - Each request is routed to Pod handling the least amount of open connections

6 – SERVICES & ROUTES

Route (OpenShift) vs Ingress (K8S)

- Route exposes service under some known and readable address
- Under the hood Routes works on OpenShift HAProxy
- Kubernetes Ingresses are the same mechanics
- RedHat created the idea of Route to fulfill the need for exposing Services to the external world. Later the idea inspired Kubernetes community to create Ingresses.

Feature	Ingress on OpenShift	Route on OpenShift
Standard Kubernetes object	X	
External access to services	X	X
Persistent (sticky) sessions	X	X
Load-balancing strategies (e.g. round robin)	X	X
Rate-limit and throttling	X	X
IP whitelisting	X	X
TLS edge termination for improved security	X	X
TLS re-encryption for improved security		X
TLS passthrough for improved security		X
Multiple weighted backends (split traffic)		X
Generated pattern-based hostnames		X
Wildcard domains		X

6 – SERVICES & ROUTES

Route – different domain

- To use different domain you need to make sure that external DNS server has been configured to route traffic to cluster
- DNS should is configured outside the OpenShift.

HEALTH MANAGEMENT

7– HEALTH MANAGEMENT

Pod logs diagnosis

- We can check logs produced by containers inside a Pod (written to stdout)
- The same as `docker logs`

7– HEALTH MANAGEMENT

Lab: Diagnosis of a Pod logs

- Lab 22
- Check YAML files
  - How they configure whole application
  - How it's different from YAML file visible in OpenShift (e.g.: on Pod page)
  - https://docs.openshift.com/container-platform/3.7/rest_api/examples.html

7– HEALTH MANAGEMENT

Health Checks

- The application must have an implemented endpoint, the answers to OpenShift health checks
- Readiness Probe
  - Whether the container running on Pod is ready to handle incoming requests
  - Works all the time
- Liveness Health Check
  - Checks if the container is alive and responds
  - OpenShift reboots pod (according to the restrat strategy)
  - Works all the time
- Startup Probe
  - Determines whether the application on the container has started
  - If Startup Probe fails, OpenShift restarts Pod
  - If configured, it runs before any other Probe

7– HEALTH MANAGEMENT

Lab: Crashing Pods

- Lab 23
- Check YAML
- In this lab we'll intentionally crash application (cause an unhandled exception). We expect OpenShift to self-heal from a failure of a Pod.
- We will also programmatically turn off Liveness Probes
  - Simulation of a situation where the application is running but does not respond to requests
  - After 3 errors in responding to the Liveness Probe, the Pod is eliminated and set up again.

STORAGE

8– STORAGE

Typy storage - ephemeral

- Ephemeral
  - Required by Pods and containers for their operational work (files, caching, logs)
  - It exists only for the duration of Pod
  - It cannot be shared
  - The space available on the cluster is shared between all Pods and other services inside OpenShift that would need it
  - OpenShift makes sure that Pods are placed on the node that has space available

8– STORAGE

Typy storage - persistent

- The type of storage that keeps saves data permanently
- OpenShift uses the K8S - Persistent Volume (PV) mechanism
- Developers make requests (Persistent Volume Claims (PVC)) to gain access to resources
  - They do not have to take into account the type of infrastructure the resources come from (NFS, cloud-provider storage)
- How Persistent Storage works
  - Resources are added to the cluster and are available for each project within the cluster - we call it Persistent Volume Resources
  - Each project can claim any number of resources in the form of Persistent Volume Claims
  - Resources obtained as a Persistent Volume Claim can be used as a Persistent Storage by Pods
- PVs are not subject to any project, they are shared between the entire cluster.
- After PV is allocated to PVC, PV cannot be used in other PVC. As a result, the PV is allocated to a specific project and cannot be used in another.
- PVs can be dynamically allocated by OpenShift
- PV is a way of managing the available space by Cluster Administrators

8– STORAGE

Lab: Save file to Persistent Storage

- Lab 24
- The file will be stored on Persistent Volume even between crashes and Pod scaling.

SECRETS, CONFIG MAPS, ENV VARS

9– SECRETS, CONFIG MAPS, ENV VARS

What they are?

- Many applications need to use:
  - some form of configuration files (JSON, XML)
  - secrets
  - environment variables
- The idea is to separate the image from these values to maintain their flexibility (e.g. easy transfer between environments)

9– SECRETS, CONFIG MAPS, ENV VARS

What is it?

- Mechanism for injecting configuration files and secrets into containers
  - The mechanism uses Persistent Volumes, so from the container point of view it is independent of OpenShift
- ConfigMap is usually whole configuration files
  - Similar to secrets, however, the purpose is to hold information that is not confidential
- Secrets can be configured to dictate the secret format (e.g. SSH auth secret, Basic Authentication).
  - By default, type = opaque so there is no type validation
- When the value is modified, it is not changed dynamically - you need to delete the previous Pod and create a new one.

9– SECRETS, CONFIG MAPS, ENV VARS

Lab: add config map to an application

- Lab 25 – config maps, secrets and env vars

AUTOSCALING

10 – AUTOSCALING

Horizontal Pod Autoscaler (HPA)

- A way to determine how OpenShift should scale up and down based on metrics (CPU or memory usage)
- Defined by specifying the minimum and maximum number of feeds that can run
- Once HPA is created, OpenShift starts checking metrics regularly and monitors resource usage
- For better HPA operation, an endpoint corresponding to the Readiness Probe should be added in the application
- HPA can be created for Deployment / Deployment Configuration and Replica Set / Replication Controller

10 – AUTOSCALING

Lab: scale application up and down

- Lab 06 - scaling

CONTACT US

In case of any questions feel free to message me:

Maciej Borowy

E-mail:
maciej.borowy@chmurowisko.pl

Websites:

www.chmurowisko.pl

www.cloudstate.eu