NobleProg

OpenShift Github Training
Advanced

Trainer: Alexander Kolin

NobleProg

Course objectives

Course objectives

Day 1

- Openshift Architecture
- Network
- Storage
- CRD & Operators
- Kustomize

Course objectives

Day 1

- Openshift Architecture
- Network
- Storage
- CRD & Operators
- Kustomize

Day 2

- Github Actions
- Pod Security
- Storage management
- Advanced Management
- DR & HA Strategies

NobleProg

Course objectives

Day 1

- Openshift Architecture
- Network
- Storage
- CRD & Operators
- Kustomize

Day 2

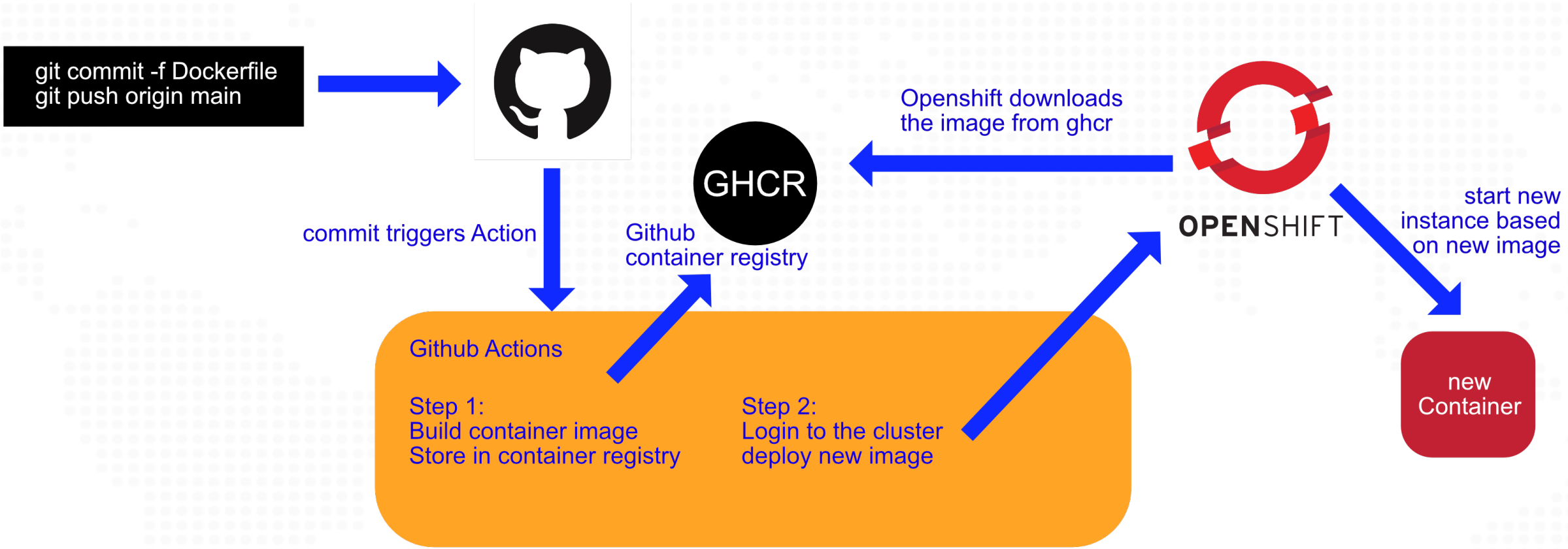
- Github Actions
- Pod Security
- Storage management
- Advanced Management
- DR & HA Strategies

Day 3

- App & Infrastructure Security
- Monitoring
- DR & HA Velero
- Wrap Up & Questions

NobleProg

Github



NobleProg

Github Actions Example

```yaml
name: Simple Echo and List Workflow with Git Pull

on:
  push:
    branches:
      - main

jobs:
  echo-and-list-job:
    runs-on: ubuntu-latest
    steps:
      - name: Set up Git
        run: |
          sudo apt-get update
          sudo apt-get install -y git

      - name: Clone Repository using Git
        run: |
          git clone https://github.com/${{ github.repository }}.git
          cd $(basename ${{ github.repository }})
          git pull

      - name: Echo Hello World
        run: echo "Hello, World!"

      - name: List Files
        run: ls *
```

NobleProg

Github Actions Example

```yaml
name: Simple Echo and List Workflow with Git Pull

on:
  push:
    branches:
      - main

jobs:
  echo-and-list-job:
    runs-on: ubuntu-latest
    steps:
      - name: Set up Git
        run: |
          sudo apt-get update
          sudo apt-get install -y git

      - name: Clone Repository using Git
        run: |
          git clone https://github.com/${{ github.repository }}.git
          cd $(basename ${{ github.repository }})
          git pull

      - name: Echo Hello World
        run: echo "Hello, World!"

      - name: List Files
        run: ls *
```

```yaml
name: Simple Echo and List Workflow

on:
  push:
    branches:
      - main

jobs:
  echo-and-list-job:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v4

      - name: Echo Hello World
        run: echo "Hello, World!"

      - name: List Files
        run: ls *
```

NobleProg

Github Actions Example

```yaml
name: Simple Echo and List Workflow with Git Pull

on:
  push:
    branches:
      - main

jobs:
  echo-and-list-job:
    runs-on: ubuntu-latest
    steps:
      - name: Set up Git
        run: |
          sudo apt-get update
          sudo apt-get install -y git

      - name: Clone Repository using Git
        run: |
          git clone https://github.com/${{ github.repository }}.git
          cd $(basename ${{ github.repository }})
          git pull

      - name: Echo Hello World
        run: echo "Hello, World!"

      - name: List Files
        run: ls *
```

```yaml
name: Simple Echo and List Workflow

on:
  push:
    branches:
      - main

jobs:
  echo-and-list-job:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v4

      - name: Echo Hello World
        run: echo "Hello, World!"

      - name: List Files
        run: ls *
```

https://github.com/marketplace

NobleProg

Actions Example Lab

```
git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/0.workflow
```

- Actions -> new Workflow -> set up a workflow
- Copy & paste helloworld.yaml
- Commit

Actions Example Lab

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/0.workflow

- Actions -> new Workflow -> set up a workflow
- Copy & paste helloworld.yaml
- Commit

```yaml
name: Hello World

on:
  push:
    branches:
      - main

jobs:
  hello:
    runs-on: ubuntu-latest

    steps:
      - name: Checkout Repository
        uses: actions/checkout@v4

      - name: Say Hello
        run: echo "Hello, World!"

      - name: list files
        run: ls -a
```

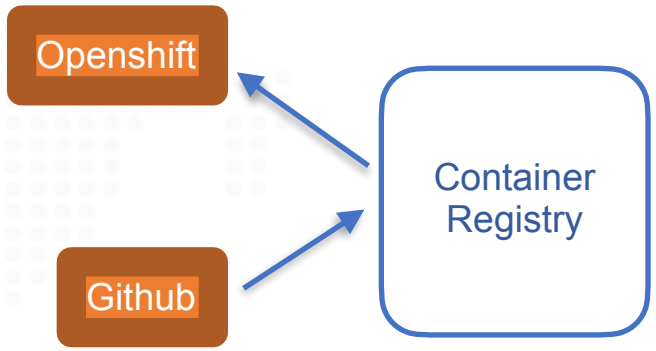
NobleProg

Github Actions Container Registry

- Github Container Registry -> create Github Packages-Token
  - User-Settings -> Developer -> PAT -> Classic

```
oc create secret docker-registry ghcr-pull-secret \
  --docker-server=ghcr.io \
  --docker-username=GITHUB_USERNAME \
  --docker-password=GITHUB_TOKEN \
  --docker-email=GITHUB_EMAIL
```

- Create Repository Secret:
  - Go to GitHub repository -> Settings -> Secrets and Variables „Actions“
    - GHCR_TOKEN



NobleProg

Github Actions vs Openshift

- Login Openshift https://developers.redhat.com/developer-sandbox
- Go to Openshift dashboard, click on your user right on top „copy login command“
  - Copy the data
  - Go to GitHub repository -> Settings -> Secrets and Variables „Actions“

  - Create Repository Secrets:
    - OPENSHIFT_SERVER - Copy the string from oc login command at „server“
    - OPENSHIFT_NAMESPACE - its your Openshift project
    - OPENSHIFT_TOKEN - Copy the string at „Your API token is“

NobleProg

Actions Container Image Lab

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/0.workflow

- Actions -> new Workflow -> set up a workflow
- Copy & paste dockerbasic.yaml
- Set correct Dockerfile path:
  e.g. ./Day1/container/Dockerfile
- Commit

Actions Container Image Lab

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/0.workflow

- Actions -> new Workflow -> set up a workflow
- Copy & paste dockerbasic.yaml
- Set correct Dockerfile path:
  e.g. ./Day1/container/Dockerfile
- Commit

```
name: Dockerbasic

env:
…
jobs:

  docker-basic:
    name: Build Dockerimage
    runs-on: ubuntu-latest
    environment: production

    …
    - name: Build from Dockerfile
      id: build-image
      uses: redhat-actions/buildah-build@v2
      with:
        image: ${{ env.APP_NAME }}
        tags: ${{ env.IMAGE_TAGS }}

        dockerfiles: |
          ./Dockerfile
    …
```

NobleProg

Actions deployment Lab

```
git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/0.workflow
```

- Actions -> new Workflow -> set up a workflow
- Copy & paste openshift.yaml
- Commit

NobleProg

Actions deployment Lab

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/0.workflow

- Actions -> new Workflow -> set up a workflow
- Copy & paste openshift.yaml
- Commit

```
...
  - name: my deployment
      run: |
        export IMAGE=${{ steps.push-image.outputs.registry-path }}
        export OPENSHIFT_NAMESPACE=${{ env.OPENSHIFT_NAMESPACE }}

        echo ${{ steps.push-image.outputs.registry-path }}

        sed -i 's|\IMAGE_PLACEHOLDER|'"$IMAGE"'|g' Day2/deployment/deployment.yaml

        oc apply -f Day2/deployment/deployment.yaml -n $OPENSHIFT_NAMESPACE
        oc apply -f Day1/network/service.yaml -n $OPENSGIFT_NAMSPACE

        oc get pods
        oc get svc
...
```

NobleProg

Security ~~PSP~~ vs PSS vs SCC

- PSP - Pod Security Policy - outdated
- Pod Security Standards
  - K8s native
  - Based on Namespace
- SCC
  - Openshift
  - Based on RBAC + Service Account
  - detailed, performant

SCC
Security Context Constraints

- SCC is an **OpenShift-specific** security feature used to control permissions and restrictions for Pods.
- It defines **what a Pod is allowed to do** in terms of security settings.
- Unlike Kubernetes **PodSecurityPolicies (PSP)**, SCCs are **assigned to ServiceAccounts**, not Pods directly.

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/1.security

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: restricted-scc
allowPrivilegedContainer: false
runAsUser:
  type: MustRunAsNonRoot
seLinuxContext:
  type: MustRunAs
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
readOnlyRootFilesystem: true
volumes:
  - configMap
  - emptyDir
  - projected
  - persistentVolumeClaim
users:
  - system:serviceaccount:default:secure-sa
```

NobleProg

SCC - Lab

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/1.security

- „Theoretically“ -> On CRC / Sandbox is not possible to test :/

Just go through README.md

```
apiVersion: security.openshift.io/v1
kind: SecurityContextConstraints
metadata:
  name: restricted-scc
allowPrivilegedContainer: false
runAsUser:
  type: MustRunAsNonRoot
seLinuxContext:
  type: MustRunAs
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
readOnlyRootFilesystem: true
volumes:
  - configMap
  - emptyDir
  - projected
  - persistentVolumeClaim
users:
  - system:serviceaccount:default:secure-sa
```

NobleProg

Storage Part 2

Perfomance Testing with FIO Pods

- FIO (**Flexible I/O Tester**) is a benchmarking tool for measuring storage performance.
- Simulates different workloads (random read/write, sequential access, mixed loads).
- Helps identify bottlenecks in storage solutions

Why Use FIO in OpenShift?

- **Test Persistent Volumes (PVs)** before deploying databases or high-IO workloads.
- **Analyze Storage Performance** under real-world conditions.
- **Compare different StorageClasses** to choose the best backend.

Storage Part 2

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/2.storage

Deployment & benchmark

- See README.md
- oc exec -it fio-test -- sh
- fio --name=randwrite --ioengine=libaio --rw=randwrite --bs=4k --size=1G --numjobs=4 -
  group_reporting

```
apiVersion: v1
kind: Pod
metadata:
  name: fio-test
spec:
  containers:
  - name: fio-container
    image: nixery.dev/shell/fio
    command: [ "sleep", "3600" ]
    volumeMounts:
    - mountPath: "/data"
      name: storage-volume
  volumes:
  - name: storage-volume
    persistentVolumeClaim:
      claimName: pvc-example
```

Understanding FIO Metrics

IOPS: Number of input/output operations per second
Latency: Response time for read/write operations
Bandwidth: Data transfer speed (MB/s)
CPU Utilization: CPU overhead during storage operations

NobleProg

Storage Part 2

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/2.storage

Practical Use Cases

- Evaluate storage performance before database deployments.
- Troubleshoot slow applications due to storage bottlenecks.
- Optimize OpenShift storage configurations.

```
apiVersion: v1
kind: Pod
metadata:
  name: fio-test
spec:
  containers:
  - name: fio-container
    image: nixery.dev/shell/fio
    command: [ "sleep", "3600" ]
    volumeMounts:
    - mountPath: "/data"
      name: storage-volume
  volumes:
  - name: storage-volume
    persistentVolumeClaim:
      claimName: pvc-example
```

NobleProg

Storage Part 2

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/2.storage

Metrics and Alerts

- Not possible on CRC / Sandbox :/

Environment Management

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/3.Management

Using Namespaces as separation of the environments

- **Better resource isolation**
- **Access control**
- **Network security**
- **Hardware separation**

NobleProg

Environment Management

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/3.Management

MySQL Deployment

- Create namespaces/projects:
  - oc create namespace dev
  - oc create namespace test
  - oc create namespace prod
- Deploy kustomize
  - cd openshift-adv/Day2/3.Management/kustomize
  - oc apply -k overlays/dev ../test ../prod

NobleProg

Environment Management Lab

git clone https://github.com/kolinrr/openshift-adv.git

cd openshift-adv/Day2/3.Management

Let's move to README.md

Disaster Recovery High Availability

Velero is

- An open-source tool for backup and restoring Kubernetes cluster resources.
- Supports on-premises and cloud environments.
- Ensures business continuity in case of failures.
- Helps in cluster migrations and rollback strategies.

Disaster Recovery High Availability

What is DR?

- **The ability to restore applications/cluster and data after a failure**
- **Common Causes:**
  - Hardware failures
  - Human errors
  - Security breaches (e.g., ransomware)
  - Natural disasters

Disaster Recovery
High Availability

What can Velero do?!

- **Backup entire K8s cluster resources, including volumes data.**
- **Restore application to another cluster**
- **Ensures quick recovery of failed services**
- **Allows cross-region or multi-cluster synch**
- **Provides point-in-time restores**

Disaster Recovery High Availability

Limitations

- **Restores can be slow**
- **Problem with restoring the stateful apps**
- **Backup of async Apps**
- **EOF-Error**

Disaster Recovery
High Availability

Best practices for HA

- **Use multi-zone (geo) or multi cluster deployments**
- **Test the backup process**
- **Test the restore process**
- **Monitor backup failures**
- **Hold „hot“-backup system**
- **pause the DB while backup**

NobleProg

Questions?

NobleProg