1 CAPA

Generated on: 2026-01-13 15:04:52 **Version:** 1.0

1.1 Table of Contents

1.2 Overview

The **Certified Argo Project Associate (CAPA)** exam demonstrates knowledge of GitOps principles and the Argo project ecosystem.

1.3 Exam Overview

Detail	Information
Exam Format	Multiple Choice
Number of Questions	60
Duration	90 minutes
Passing Score	75%
Certification Validity	3 years
Cost	$250 USD
Retake Policy	1 free retake

1.4 Exam Domains & Weights

Domain	Weight
Argo CD	40%
Argo Workflows	25%
Argo Rollouts	20%
Argo Events	15%

1.5 Key Topics

1.5.1 Argo CD

- Application deployment and sync
- Repository management
- Application health and sync status
- Multi-cluster deployment

1.5.2 Argo Workflows

- Workflow templates
- DAG and steps
- Artifacts and parameters
- Workflow execution

1.5.3 Argo Rollouts

- Blue-green deployments
- Canary deployments
- Analysis and metrics
- Traffic management

1.5.4 Argo Events

- Event sources
- Sensors and triggers
- Event-driven workflows

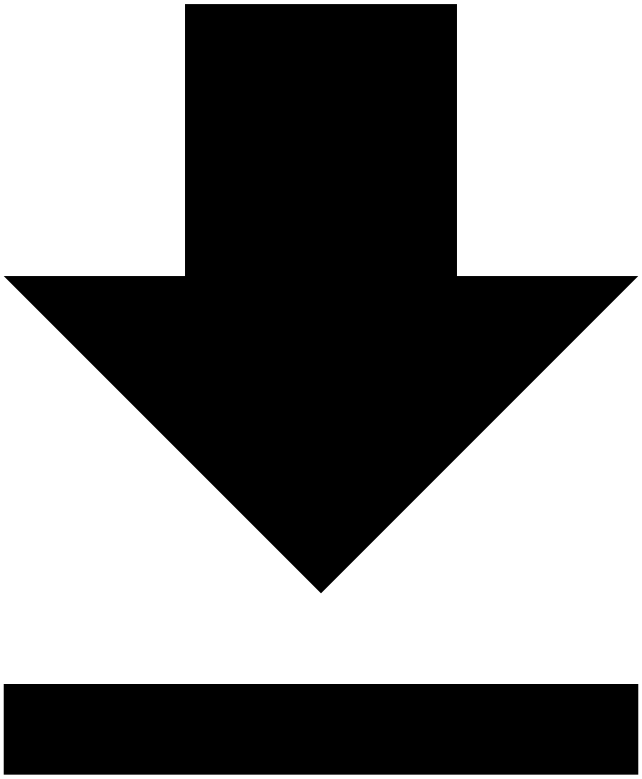
1.6 Study Resources

- Argo CD Documentation
- Argo Workflows Documentation
- Argo Rollouts Documentation
- CAPA Curriculum

1.7 Navigation

- Next: Sample Questions →

1.8 Argo CD Fundamentals



Download PDF Version

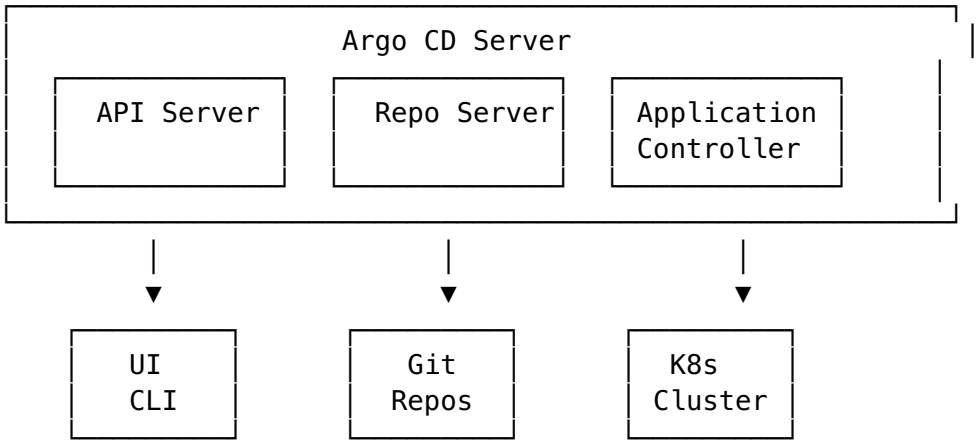
Comprehensive guide to Argo CD for CAPA certification.

1.9 Overview

Argo CD is a declarative GitOps continuous delivery tool for Kubernetes:

- **Declarative** - Application definitions in Git
- **Automated** - Sync applications automatically
- **Auditable** - Git history as audit log
- **Multi-cluster** - Manage multiple clusters

1.10 Architecture



1.10.1 Components

- **API Server** - gRPC/REST API, Web UI, CLI
- **Repository Server** - Clones Git repos, generates manifests
- **Application Controller** - Monitors applications, syncs state

---

1.11 Installation

```
# Create namespace
kubectl create namespace argocd

# Install Argo CD
kubectl apply -n argocd -f https://raw.githubusercontent.com/
        argoproj/argo-cd/stable/manifests/install.yaml

# Get admin password
kubectl -n argocd get secret argocd-initial-admin-secret -o
        jsonpath="{.data.password}" | base64 -d

# Port forward UI
kubectl port-forward svc/argocd-server -n argocd 8080:443

# Install CLI
curl -sSL -o argocd https://github.com/argoproj/argo-cd/releases/
        latest/download/argocd-linux-amd64
chmod +x argocd
sudo mv argocd /usr/local/bin/

# Login
argocd login localhost:8080
```

1.12 Application Resource

1.12.1 Basic Application

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: guestbook
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/argoproj/argocd-example-apps.git
    targetRevision: HEAD
    path: guestbook
  destination:
    server: https://kubernetes.default.svc
    namespace: guestbook
```

1.12.2 Application with Helm

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: nginx
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://charts.bitnami.com/bitnami
    chart: nginx
    targetRevision: 15.0.0
    helm:
      values: |
        replicaCount: 2
        service:
          type: ClusterIP
  destination:
    server: https://kubernetes.default.svc
    namespace: nginx
```

1.12.3 Application with Kustomize

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: kustomize-app
  namespace: argocd
spec:
  project: default
```

```yaml
  source:
    repoURL: https://github.com/myorg/myapp.git
    targetRevision: HEAD
    path: overlays/production
    kustomize:
      images:
      - myapp=myregistry/myapp:v1.0.0
  destination:
    server: https://kubernetes.default.svc
    namespace: production
```

1.13 Sync Policies

1.13.1 Automated Sync

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: auto-sync-app
spec:
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
      allowEmpty: false
    syncOptions:
    - CreateNamespace=true
    - PruneLast=true
    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m
```

1.13.2 Sync Options

Option	Description
`prune`	Delete resources not in Git
`selfHeal`	Auto-sync when drift detected
`CreateNamespace`	Create namespace if missing
`PruneLast`	Prune after sync
`ApplyOutOfSyncOnly`	Only sync out-of-sync resources

1.14 Projects

```yaml
apiVersion: argoproj.io/v1alpha1
kind: AppProject
metadata:
  name: production
  namespace: argocd
spec:
  description: Production applications
  sourceRepos:
  - 'https://github.com/myorg/*'
  destinations:
  - namespace: 'prod-*'
    server: https://kubernetes.default.svc
  clusterResourceWhitelist:
  - group: ''
    kind: Namespace
  namespaceResourceWhitelist:
  - group: '*'
    kind: '*'
  roles:
  - name: developer
    policies:
    - p, proj:production:developer, applications, get, production/
        *, allow
    - p, proj:production:developer, applications, sync,
        production/*, allow
```

1.15 ApplicationSet

```yaml
apiVersion: argoproj.io/v1alpha1
kind: ApplicationSet
metadata:
  name: cluster-apps
  namespace: argocd
spec:
  generators:
  - list:
      elements:
      - cluster: dev
        url: https://dev.example.com
      - cluster: staging
        url: https://staging.example.com
      - cluster: prod
        url: https://prod.example.com
  template:
    metadata:
      name: '{{cluster}}-app'
    spec:
      project: default
```

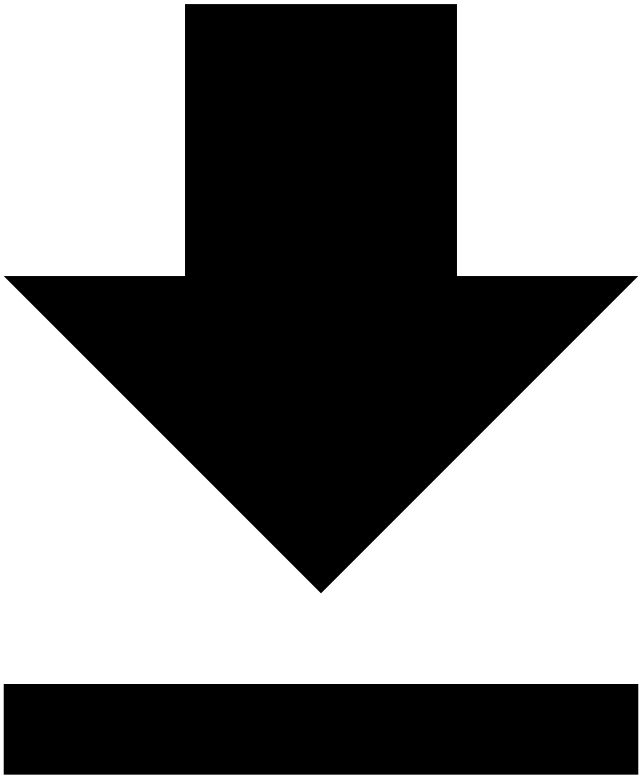
```yaml
    source:
      repoURL: https://github.com/myorg/myapp.git
      targetRevision: HEAD
      path: 'envs/{{cluster}}'
    destination:
      server: '{{url}}'
      namespace: myapp
```

1.16 CLI Commands

```
# Application management
argocd app list
argocd app get <app-name>
argocd app create <app-name> --repo <repo> --path <path> --dest-
        server <server> --dest-namespace <ns>
argocd app sync <app-name>
argocd app delete <app-name>

# Sync operations
argocd app sync <app-name> --prune
argocd app sync <app-name> --force
argocd app diff <app-name>

# History and rollback
argocd app history <app-name>
argocd app rollback <app-name> <revision>

# Project management
argocd proj list
argocd proj get <project-name>
```

← Back to CAPA Overview

1.17 Sample Practice Questions



Download PDF Version

1.18 Practice Resources

- Argo CD Documentation
- Killercoda Argo Scenarios

1.19 Argo CD (40%)

1.19.1 Question 1

Create an Argo CD Application that deploys from a Git repository.

Show Solution

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: my-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://github.com/org/repo.git
    targetRevision: HEAD
    path: manifests
  destination:
    server: https://kubernetes.default.svc
    namespace: default
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```

1.19.2 Question 2

How do you sync an application manually using the CLI?

Show Solution

```
argocd app sync my-app

# With prune
argocd app sync my-app --prune

# Force sync
argocd app sync my-app --force
```

1.19.3 Question 3

Configure an Application to use a Helm chart with custom values.

Show Solution

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: helm-app
  namespace: argocd
spec:
  project: default
  source:
    repoURL: https://charts.example.com
    chart: my-chart
    targetRevision: 1.0.0
    helm:
      values: |
```

```
        replicaCount: 3
        image:
          tag: latest
      parameters:
      - name: service.type
        value: LoadBalancer
  destination:
    server: https://kubernetes.default.svc
    namespace: default
```

1.19.4 Question 4

What are the different sync statuses in Argo CD?

Show Solution

Sync Status: - **Synced** - Live state matches desired state - **OutOfSync** - Live state differs from desired state - **Unknown** - Unable to determine sync status

Health Status: - **Healthy** - Resource is healthy - **Progressing** - Resource is not healthy but could become healthy - **Degraded** - Resource is degraded - **Suspended** - Resource is suspended - **Missing** - Resource does not exist - **Unknown** - Health status unknown

---

1.20 Argo Workflows (25%)

1.20.1 Question 5

Create a simple Argo Workflow with two sequential steps.

Show Solution

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: hello-world-
spec:
  entrypoint: main
  templates:
  - name: main
    steps:
    - - name: step1
        template: hello
    - - name: step2
        template: world
  - name: hello
    container:
      image: alpine
      command: [echo, "Hello"]
  - name: world
```

```
    container:
      image: alpine
      command: [echo, "World"]
```

1.20.2 Question 6

Create a DAG workflow with parallel tasks.

Show Solution

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: dag-workflow-
spec:
  entrypoint: main
  templates:
  - name: main
    dag:
      tasks:
      - name: A
        template: task
      - name: B
        template: task
        dependencies: [A]
      - name: C
        template: task
        dependencies: [A]
      - name: D
        template: task
        dependencies: [B, C]
  - name: task
    container:
      image: alpine
      command: [echo, "Running task"]
```

1.20.3 Question 7

Pass parameters between workflow steps.

Show Solution

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: param-workflow-
spec:
  entrypoint: main
  templates:
  - name: main
    steps:
    - - name: generate
```

```
        template: gen-output
    - - name: consume
        template: use-output
        arguments:
          parameters:
          - name: message
            value: "{{steps.generate.outputs.parameters.result}}"
  - name: gen-output
    container:
      image: alpine
      command: [sh, -c, "echo 'Hello' > /tmp/result.txt"]
    outputs:
      parameters:
      - name: result
        valueFrom:
          path: /tmp/result.txt
  - name: use-output
    inputs:
      parameters:
      - name: message
    container:
      image: alpine
      command: [echo, "{{inputs.parameters.message}}"]
```

1.21 Argo Rollouts (20%)

1.21.1 Question 8

Create a Canary rollout that shifts 20% traffic initially.

Show Solution

```
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: canary-rollout
spec:
  replicas: 5
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: app
        image: my-app:v2
  strategy:
```

```yaml
    canary:
      steps:
      - setWeight: 20
      - pause: {duration: 1m}
      - setWeight: 50
      - pause: {duration: 1m}
      - setWeight: 80
      - pause: {duration: 1m}
```

1.21.2 Question 9

Create a Blue-Green rollout.

Show Solution

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Rollout
metadata:
  name: bluegreen-rollout
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: app
        image: my-app:v2
  strategy:
    blueGreen:
      activeService: my-app-active
      previewService: my-app-preview
      autoPromotionEnabled: false
```

1.21.3 Question 10

Promote or abort a rollout using kubectl.

Show Solution

```bash
# Promote rollout
kubectl argo rollouts promote my-rollout

# Abort rollout
kubectl argo rollouts abort my-rollout

# Retry rollout
kubectl argo rollouts retry my-rollout
```

```
# Check status
kubectl argo rollouts status my-rollout
```

1.22 Argo Events (15%)

1.22.1 Question 11

Create an EventSource for GitHub webhooks.

Show Solution

```yaml
apiVersion: argoproj.io/v1alpha1
kind: EventSource
metadata:
  name: github-eventsource
spec:
  github:
    example:
      repositories:
      - owner: my-org
        names:
        - my-repo
      webhook:
        endpoint: /push
        port: "12000"
        method: POST
      events:
      - push
      apiToken:
        name: github-token
        key: token
```

1.22.2 Question 12

Create a Sensor that triggers a workflow on events.

Show Solution

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Sensor
metadata:
  name: github-sensor
spec:
  dependencies:
  - name: github-dep
    eventSourceName: github-eventsource
    eventName: example
  triggers:
  - template:
```

```yaml
    name: workflow-trigger
    k8s:
      operation: create
      source:
        resource:
          apiVersion: argoproj.io/v1alpha1
          kind: Workflow
          metadata:
            generateName: github-workflow-
          spec:
            entrypoint: main
            templates:
            - name: main
              container:
                image: alpine
                command: [echo, "Triggered by GitHub"]
```

1.23 Exam Tips

1. **Know Argo CD Application spec** - source, destination, syncPolicy
2. **Understand sync strategies** - automated vs manual, prune, selfHeal
3. **Practice Workflow templates** - steps, DAG, parameters, artifacts
4. **Know Rollout strategies** - canary steps, blue-green services
5. **Understand event-driven architecture** - EventSource, Sensor, Trigger

---

← Back to CAPA Overview
