



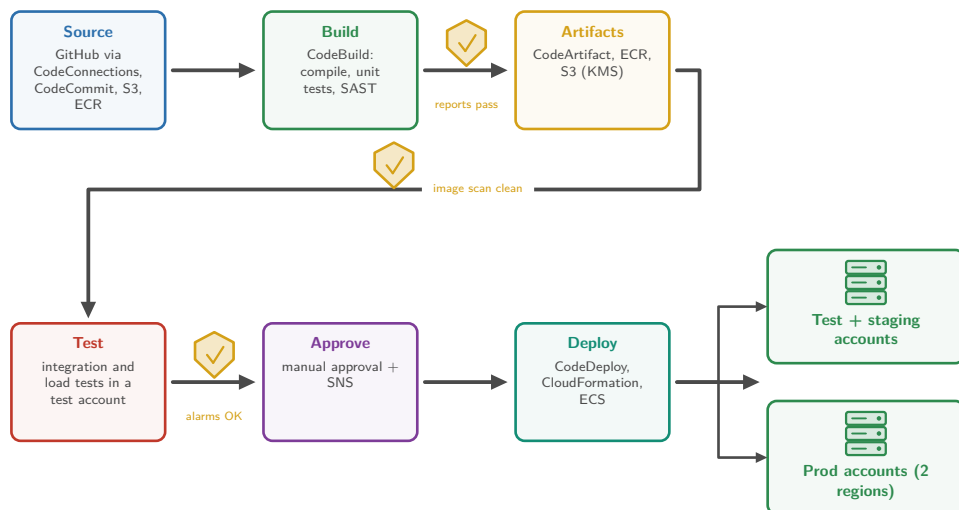
DOP-C02 Domain 1: SDLC Automation and Pipelines

AWS Certified DevOps Engineer - Professional, Domain 1 (22 %) | CodePipeline V2, CodeBuild, CodeArtifact, ECR, test stages, cross-account delivery; the DVA-C02 chapter teaches the basics

DOP-C02 Domain 1 CodePipeline CodeBuild CI/CD

© What Domain 1 asks you to do

Task statement (summary)	What the exam expects	Services
1.1 Implement CI/CD pipelines	source triggers, build, test, approval and deploy stages; multi-account and multi-region delivery; secrets in builds	CodePipeline, CodeBuild, CodeConnections, CodeCommit, S3, KMS
1.2 Integrate automated testing	unit, integration, load, security tests at the right stage; fail fast; report results	CodeBuild reports, Device Farm, Inspector, Lambda test actions
1.3 Build and manage artifacts	versioned, immutable artifacts; package repositories; image registries and lifecycle	CodeArtifact, ECR, S3 versioning, Image Builder
1.4 Deployment strategies	in-place, rolling, blue/green, canary on EC2, ECS, Lambda (sheet 02)	CodeDeploy, ECS, Elastic Beanstalk, SAM

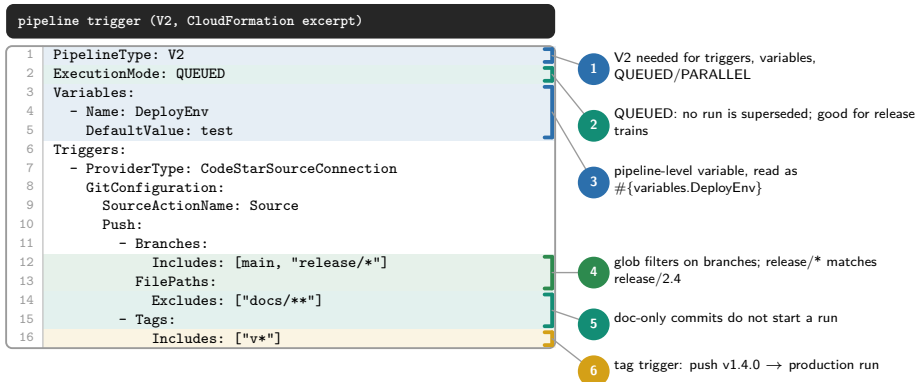


One pipeline in a tooling account delivering to several accounts and regions

✳ CodePipeline V2 in depth

V1 pipeline	V2 pipeline (default for new pipelines)
triggers: any push on the configured branch	triggers on push or pull request with branch, file-path and Git tag filters (include / exclude, glob)
execution mode SUPERSEDED only	SUPERSEDED, QUEUED or PARALLEL
action variables only (namespaces)	pipeline-level variables set at start (--variables name=k,value=v) plus action output variables
retry a failed stage by hand	stage conditions (entry, on success, on failure) and automatic rollback of a failed stage to the last successful execution
billed per active pipeline per month	billed per action-execution minute (check the pricing page)

Action category	Providers worth knowing	Exam hook
Source	CodeConnections (GitHub, GitHub Enterprise, GitLab, Bitbucket), CodeCommit, S3 (versioned bucket), ECR	S3 and ECR sources start the pipeline through EventBridge events
Build / Test	CodeBuild, Jenkins (plugin), Commands action (shell commands on CodeBuild compute, V2)	Commands = quick scripts without a separate CodeBuild project
Deploy	CodeDeploy, CloudFormation (stacks, change sets, StackSets), ECS (rolling or blue/green), Elastic Beanstalk, S3, Service Catalog, App-Config, Amazon EKS	CloudFormation action modes: CREATE_UPDATE, CHANGE_SET_REPLACE then CHANGE_SET_EXECUTE
Approval	Manual approval with SNS topic, URL to review, comments	times out after 7 days → stage fails
Invoke	Lambda (must call PutJobSuccessResult / PutJobFailureResult), Step Functions	a Lambda that forgets to report leaves the action running until time-out



EXAM TIP

"Only start the pipeline when files under `src/` change" or "deploy to prod when a Git tag is pushed" → CodePipeline **V2 trigger filters**, not an EventBridge rule plus Lambda. "Several commits arrive during a 40-minute run and each must be deployed in order" → **QUEUED** mode; **SUPERSEDED** would skip the waiting middle one.

CodeBuild: buildspec, secrets, cache, reports

buildspec.yml (version 0.2) for a container image

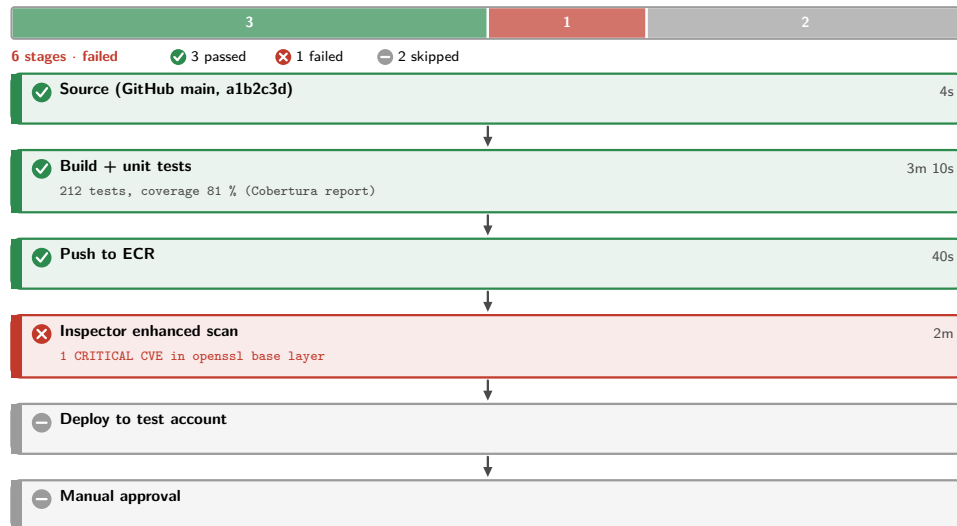
```
version: 0.2
env:
  variables: {IMAGE_REPO: "app"}
  parameter-store: {SONAR_URL: /ci/sonar/url}
  secrets-manager: {SONAR_TOKEN: ci/sonar:token}
  exported-variables: [IMAGE_TAG]
phases:
  install:
    runtime-versions: {python: 3.12}
  pre_build:
```

```
  commands:
    - aws ecr get-login-password | docker login
    --username AWS --password-stdin $REGISTRY
    - export IMAGE_TAG=${CODEBUILD_RESOLVED_SOURCE_VERSION:0:7}
  build:
    commands:
    - pytest --junitxml=reports/unit.xml
    - docker build -t $REGISTRY/$IMAGE_REPO:$IMAGE_TAG .
  post_build:
    commands:
```

```
- docker push $REGISTRY/$IMAGE_REPO:$IMAGE_TAG
- printf '{"name":"app","imageUri":"%s"}'
  $REGISTRY/$IMAGE_REPO:$IMAGE_TAG > imagedefinitions.json
reports:
  unit: {files: [unit.xml], base-directory: reports,
         file-format: JUNITXML}
artifacts: {files: [imagedefinitions.json]}
cache: {paths: ["/root/.cache/pip/**/*"]}
```

Phases	install → pre_build → build → post_build; a failure in build still runs post_build (check <code>CODEBUILD_BUILD_SUCCEEDING</code>); finally: blocks run whatever happens
Docker builds	enable privileged mode on the project; the service role needs <code>ecr:GetAuthorizationToken</code> plus push actions on the repository
Secrets	parameter-store and secrets-manager keys in env resolve at build start and are masked in logs; never put secrets in plaintext variables
Cache	S3 cache (shared across hosts) or local cache modes: <code>LOCAL_SOURCE_CACHE</code> , <code>LOCAL_DOCKER_LAYER_CACHE</code> , <code>LOCAL_CUSTOM_CACHE</code> (best effort, same host)
Compute	on-demand EC2 (small to 2xlarge, GPU, ARM), Lambda compute (fast start, no Docker, 15 min max), reserved-capacity fleets; build timeout 5 min to 36 h (default 60 min)
VPC builds	attach the project to private subnets to reach RDS or internal APIs; it then needs a NAT gateway or VPC endpoints for S3, ECR, logs
Batch builds	batch: with build-list, build-matrix (OS × runtime) or build-graph (dependencies) runs several builds from one start

One run of the image pipeline, failed at the scan gate



Worked example: what caching saves in build minutes

PROBLEM A team runs $B = 40$ builds per working day, $D = 22$ working days a month. Each build takes $T_0 = 12$ min; 5 min of it is downloading dependencies. An S3 cache cuts that download to 0.5 min. How many build minutes per month are saved?

GIVEN

- $B = 40$ builds/day
- $D = 22$ days/month
- $T_0 = 12$ min per build without cache
- $T_1 = 12 - 5 + 0.5 = 7.5$ min per build with cache

1 Monthly minutes without cache

$$M_0 = B \times D \times T_0 = 40 \times 22 \times 12 = 10,560 \text{ min}$$

Why: CodeBuild bills per build minute of the chosen compute type

2 Monthly minutes with cache

$$M_1 = 40 \times 22 \times 7.5 = 6,600 \text{ min}$$

3 Saving

$$M_0 - M_1 = 3,960 \text{ min (37.5 \%)}$$

Why: same share of the bill and of the developer wait time

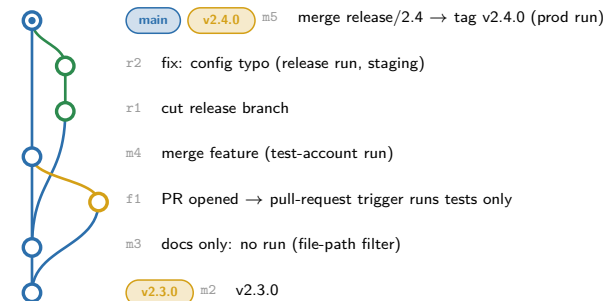
ANSWER 3,960 build minutes per month, 37.5 % less, and 4.5 min shorter feedback per commit

CHECK ✓ $7.5 / 12 = 0.625 \rightarrow \text{saving } 1 - 0.625 = 37.5 \%$

Artifacts: CodeArtifact, ECR, S3

Command	What it does
<code>aws codeartifact login --tool pip --domain acme --repository py-internal</code>	configures pip with a token (valid 12 h by default) and the repository endpoint; also <code>--tool npm, twine, dotnet, swift</code>
<code>aws codeartifact associate-external-connection --external-connection public:pypi ...</code>	lets a repository fetch from public PyPI; usually on a dedicated upstream repo
<code>aws codeartifact put-package-origin-configuration ...</code>	package origin controls: block public versions of an internal package name (dependency-confusion defence)
<code>aws ecr put-image-tag-mutability --image-tag-mutability IMMUTABLE</code>	a tag can never be overwritten: v1.4.0 always means the same digest
<code>aws ecr put-lifecycle-policy --lifecycle-policy-text file://keep30.json</code>	expire untagged images after N days, keep the last 30 tagged
<code>aws ecr put-registry-scanning-configuration --scan-type ENHANCED</code>	continuous Inspector scanning of OS and language packages instead of basic scan-on-push
<code>aws ecr put-replication-configuration ...</code>	registry-level cross-region and cross-account replication, filtered by repository prefix

Release flow that the triggers above implement



💡 Build once, promote the same artifact

Build the image or zip once, tag it with the commit SHA, then deploy that exact digest to test, staging and prod. Rebuilding per environment breaks traceability and can ship code that was never tested.

🔒 Cross-account and secure pipelines

- Customer-managed KMS key** artifact bucket encrypted with a CMK whose key policy lets the prod account's role decrypt; the default `aws/s3` key cannot be shared across accounts
- Bucket policy** allow the prod account role `s3:GetObject*` and `s3:PutObject` on the artifact bucket
- Cross-account role in prod** trusted by the tooling account; the pipeline action sets `RoleArn` to it

- 4 CloudFormation deployment role a second role in prod passed to CloudFormation (--role-arn) holding the infrastructure permissions
- 5 Per-region artifact stores a cross-region action needs an artifact bucket in that region (ArtifactStores map)

Pipeline in a tooling account deploying into a prod account

Pipeline security baseline

✓ baseline / mandatory □ recommended – verify in your context

- ✓ Source from CodeConnections (app install, scoped repos), never a personal access token in a variable
- ✓ CodeBuild service role scoped per project; no AdministratorAccess
- ✓ Secrets from Secrets Manager or Parameter Store SecureString in buildspec env
- ✓ Artifacts encrypted with a CMK; bucket blocks public access; versioning on
- MUST SAST, dependency (SCA) and image scanning gate before any deploy
- SHOULD Manual approval or alarm-based stage condition before production
- SHOULD Notifications rule (CodeStar Notifications / EventBridge) on failed executions to chat or SNS

⚠ CodeCommit status

AWS closed CodeCommit to new customers in July 2024 and later reversed course; as of 2026 check its availability before designing around it. Exam questions still use CodeCommit as a source; in the field, GitHub or GitLab through CodeConnections is the common source.

✓ Testing in the pipeline

Test type	Where in the pipeline	Blocks the release?	Tool on AWS
Unit + lint + SAST	build stage, every commit and PR	yes	CodeBuild with reports
Dependency and image scan	after build, before deploy	yes	Inspector (ECR enhanced, Lambda, code scanning)
Integration / API	test account after deploy	yes	CodeBuild, Lambda invoke action
Load / performance	staging, before prod	partial	Distributed Load Testing on AWS solution, CodeBuild
Mobile / browser	test stage	partial	Device Farm
Synthetic canary	after prod deploy, continuously	no	CloudWatch Synthetics

Myth	Fact	Why
CodePipeline runs the build itself	CodePipeline only orchestrates; CodeBuild, Jenkins or a Commands action executes	Build logs live in the CodeBuild project (CloudWatch Logs / S3), not in the pipeline.
A Lambda invoke action succeeds when the function returns	It succeeds only after PutJobSuccessResult	Without the callback the action waits until its timeout and fails.
Cross-account pipelines work with the default S3 key	They need a customer-managed KMS key shared in its key policy	The AWS-managed aws/s3 key policy cannot be edited.
Rebuilding per environment is safer	Promote one immutable artifact	Same digest in every account proves prod runs what was tested.

🚩 Recap and self-test

Key takeaways

- 01 V2 pipelines
Git triggers with filters, variables, QUEUED/PARALLEL, stage conditions, rollback
- 02 buildspec 0.2
phases, env secrets, reports, cache, exported variables
- 03 Build once, promote
immutable ECR tags, CodeArtifact for packages
- 04 Cross-account
CMK on the artifact bucket + roles in each target account
- 05 Gate with tests and scans
fail fast, report results, approve before prod

Q1 Which CodePipeline feature starts a run only for pushes touching src/**?

A V2 trigger with a file-path include filter (GitConfiguration Push FilePaths).

Q2 Commits must deploy in order without skipping. Execution mode?

A QUEUED (V2); SUPERSEDED would replace a waiting run with the newest one.

Q3 How does CodeBuild read a database password safely?

A buildspec env: secrets-manager: (or parameter-store: SecureString); values are masked in logs.

Q4 Docker build fails with 'Cannot connect to the Docker daemon'. Fix?

A Enable privileged mode on the CodeBuild project (or use a Docker-enabled image).

Q5 Which file does the ECS standard deploy action read from the build artifact?

A imagedefinitions.json (container name + image URI); blue/green via CodeDeploy uses imageDetail.json plus appspec and taskdef.

Q6 Stop developers pulling a malicious public package that copies an internal name?

A CodeArtifact package origin controls: block upstream versions for that package.

Q7 Pipeline in account A deploys to account B and fails with AccessDenied on artifacts. Typical cause?

A Artifact bucket uses the default S3 key or a key policy not granting account B's role decrypt.

Q8 How long does a manual approval wait?

A Up to 7 days, then the action fails.

Q9 Keep only the last 30 tagged images and delete untagged ones after 7 days?

A An ECR lifecycle policy with two rules (tagStatus untagged sinceImagePushed 7 days; tagged imageCountMoreThan 30).

Q10 A Lambda invoke action hangs until timeout although the code finished. Why?

A The function never called `codepipeline.put_job_success_result` (or failure).