AWS Code Build : Short Interview questions

🛠️ 1. What is AWS CodeBuild in a CI/CD Workflow?

It wakes up when CodePipeline calls or when you push your code, compiles your code, runs tests, packages it, and says, *"Here, deployment-ready!"*
No servers to manage. No queues to wait in.

📜 2. What is buildspec.yml and why is it important?

Think of buildspec.yml as a to-do list you leave for CodeBuild.
We tell CodeBuild, first install dependencies, build the app, then run tests and finally upload the zip file.
It's divided into:

- **install**
- **pre_build**
- **build**
- **post_build**

---

3. What compute types does CodeBuild offer?

- Small jobs → build.general1.small (2 vCPU, 3 GB RAM)
- Big builds → Medium, Large, 2XL
- Special needs → **ARM** (Graviton for savings), **GPU** (for ML or graphics)

Pick what suits your code — and pay only for what you use.

---

4. How are artifacts handled?

Once the build is done, *"Where should I keep this output?"*

Artifacts — your .zip, .jar, or built files — are:

- Defined in the **artifacts: block in buildspec.yml**
- Uploaded to S3 or handed to the next pipeline step
- Can be renamed, split into primary/secondary, or filtered

---

5. How do you pass secrets securely?

Secrets are like passwords.
In CodeBuild, you store them in:

- **AWS Secrets Manager**
- **SSM Parameter Store**
  Then in buildspec.yml, say:

```
secrets-manager:
  DB_PASS: "my-secret:password"
```

IAM ensures only CodeBuild can access it.

---

7. How can you make build faster ?

If your build keeps downloading the same libraries — that's wasted time.
Solution: caching!
Tell CodeBuild:
yaml

```
cache:
  paths:
    - 'node_modules/'
    - '/root/.m2/repository'
```

CodeBuild will reuse them in future runs.
Use **S3** or **local cache** for faster builds.

---

8. How to troubleshoot failed builds?

- Open **CloudWatch Logs**
- Add debug lines like env, df -h, free -m
- Check IAM permissions, VPC access, and missing tools

---

9. How to run parallel or matrix builds?

Have 5 test suites or 3 runtimes? Run them **in parallel!**

- Enable **Batch Builds** in CodeBuild
- Use dynamic variables
  Now, each set runs independently. Faster feedback, faster fixes.

---

10. How to handle secrets with Systems Manager/Secrets Manager?

Same as earlier — just with structure.

- Use parameter-store: or secrets-manager: in buildspec.yml
- IAM role should have GetParameters or GetSecretValue
- You can even rotate secrets automatically via Lambda.

💸 **11. How to reduce costs in frequent deployments?**

- Use **ARM (Graviton)** instances (20–40% cheaper!)
- Enable **caching**, minimize Docker layer rebuilds
- Use **conditional builds** (only run if code actually changed)

📊 **13. How to view test reports?**

Tests pass or fail? CodeBuild can show it — graphically.
In buildspec.yml:
yaml

```
reports:
  unit-test:
    files:
      - test-results/*.xml
    file-format: JUNITXML
```

View in console, export to S3, or gate deployment in CodePipeline.

🐢 **14. Builds are slow — how to optimize?**

Speed it up with:

- Bigger compute (for concurrency)
- Pre-cached base Docker images
- Skipping unnecessary steps
- Smart triggers (build only changed services)

---

1. What is the purpose of buildspec.yml in AWS CodeBuild?

A. To define the CI/CD pipeline stages
B. To configure IAM permissions
C. To instruct CodeBuild on steps like install, build, test, and post-build
D. To set up ECS deployment rules

✅ **Correct Answer:** C
📝 *Explanation:* buildspec.yml acts as a step-by-step instruction file for CodeBuild. It includes phases like install, pre_build, build, and post_build.

---

2. In buildspec.yml, where do you define the output .zip or .jar files?

A. In the install phase
B. In the artifacts section
C. In the post_build commands
D. In the cache section

✅ **Correct Answer:** B
📝 *Explanation:* Artifacts are declared in the artifacts: block, specifying what output files to store or pass forward.

---

3. How can you speed up builds in CodeBuild?

A. Disable CloudWatch

B. Use the smallest compute type

C. Enable caching for dependencies

D. Skip all tests

✅ **Correct Answer:** C

📝 *Explanation:* Using cache paths for libraries or dependency folders avoids downloading them on every run.
