Question 1: Explain the differences between development, integration, staging, and production environments.

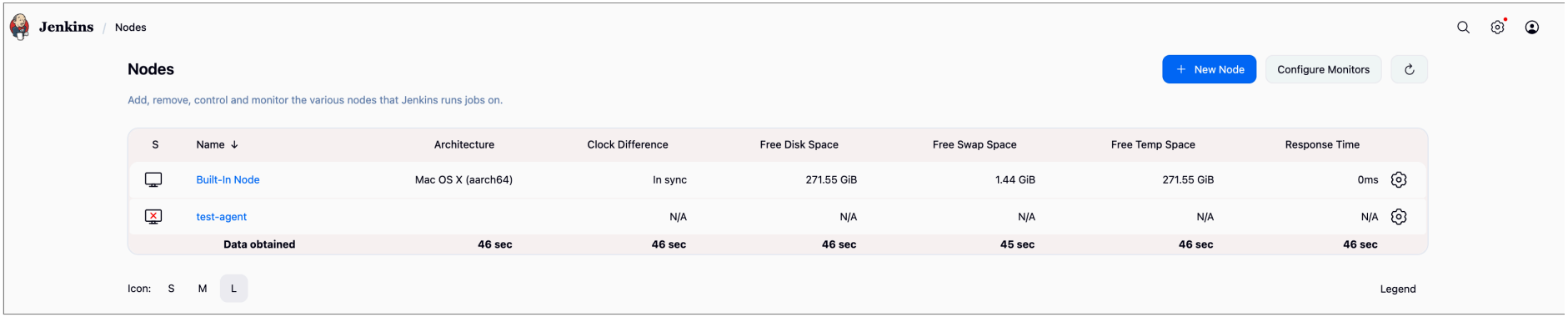
Answer:

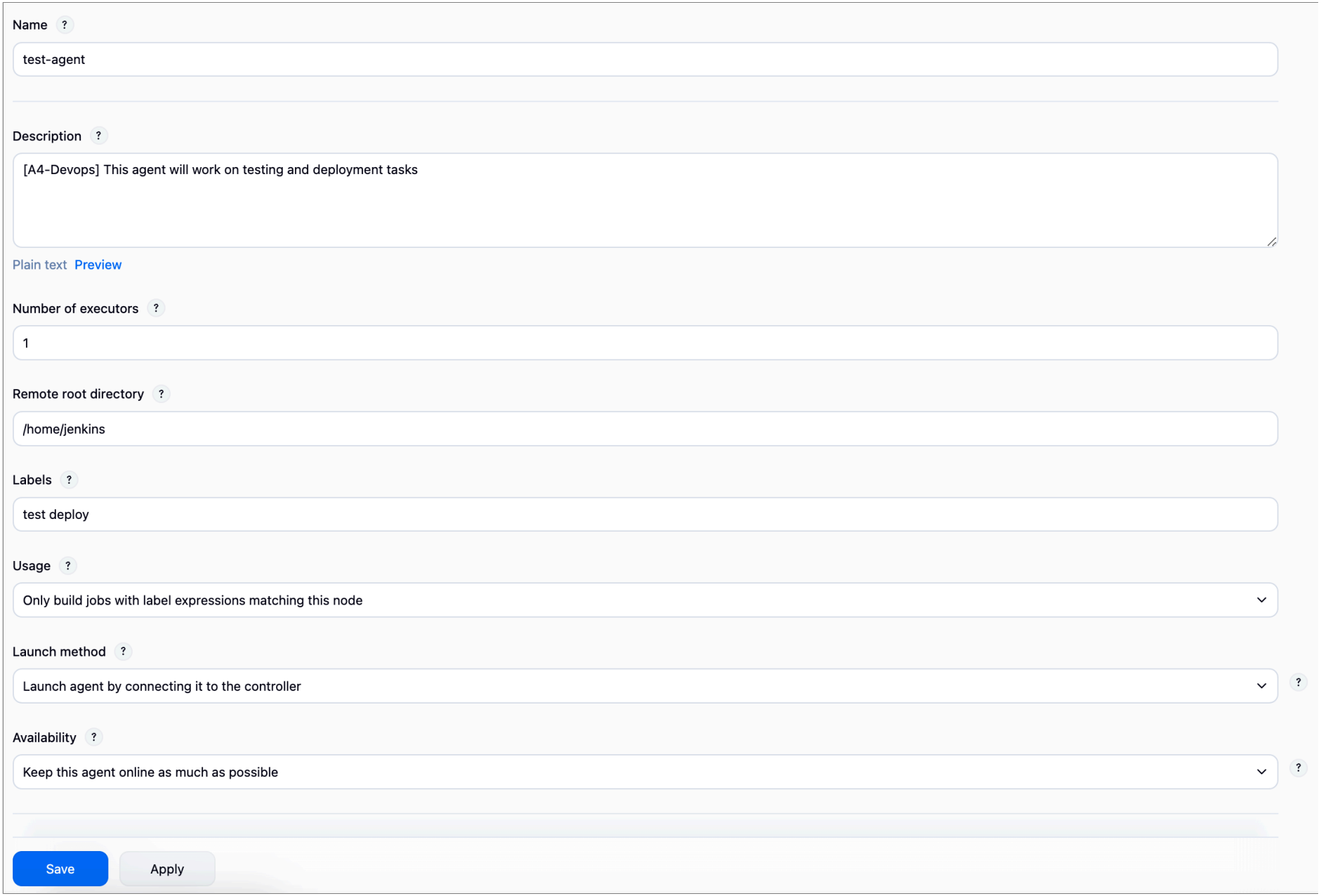
Environment	Purpose & Characteristics	Data Management Strategies	Access control & Security	Testing approaches
Development	Here is where the code starts, meaning where features of any app start being developed or features can be modified. Anyway, this means starting with new code or check out existing code from an existing branch	This stage has access to local data or *mock data* to simulate production data	Always with a version control to keep changes. This stage is still very *"local"* meaning that access here is mostly to everyone but the ones performing changes are developers. Security can be checked with statyic analyzers	Mostly local testing with unit testing (checking functional correctness. Need to apply sunny day testing, rainy day testing and regression testing), Code Quality (is code readable for other developers?) and peer review and their feedback
Integration	Is where the CI server works. This one is triggered by a commit or Pull Request to the version control system. This ensures that when someone says "my code is ready", a not local test is performed to check all standards are met before merging the new code/existing code with tweaks into the existing code.	It's always done again thorough the version control system (VCS) so it's possible to add/merge code such that there's a tracking system for versions when something breaks or doesn't work. This part can use syntethic data or parts of real data to test functional correctness.	Since this is the part we are trying to merge code upstream in the structure, access here is mostly to developers so they can check that everything they're doing is working and compliance with standars set for a staging environment	Testing here is not only for the new developed code but for all parts of the code. This is done to check that the new code didn't break any other part of the program. Basically, saying that the new version is working on it's overall. Testing needs to cover all parts, yet it can't be very large testing because we need it to be fast and have fast feedback.
Staging	Purpose of this environment is basically to simulate production environment (real life) to check that everything is working as expected and that UX is consistent. It's triggered by the clean-up of the integration environment.	Data in the environment shoud be a copy of the production database or a large part of it (again, trying to simulate production environment). When doing this part, sensitive or restricted data must be hidden.	This environment needs to check in its steps that any sensitive data is restricted to the non-target users, meaning that only people/agents with permission can access this type of data. For this part, access is only for responsibles of checking overall functionality of the app, and also, the go/no-go mechanism restricted to those who can make the decision. It has to be traceable to check any issues in the process.	Since it's trying to simulate production, all type of testing needs to be done (load, security, compliance, user) to ensure that the app is working well in all it's aspects.

Environment	Purpose & Characteristics	Data Management Strategies	Access control & Security	Testing approaches
Production	Is when the app/new features are released to they can be used by real users. It's triggered after the "Go" decision in the staging environment.	Most of the times using versions of the app so users and developers can keep track of release of features complementing the app. Data here is production, using the totality of the resources that are needed for the app to work. Again, restricted data has to be obscured to those who don't have access to it.	Production characterizes for having the code being already in the field. Meaning that security should be working completely with any restricted access working, and possibly with no security breaches.	Testing here is basically the performance of the app in real life: how are the load metrics, queries per second, etc. With the data from this performance metrics, strategic decisions can be done to start plannig for new features that can improve the existing app, meaning that this ones need to start in the development environment, completing the cycle.

Question 2: Jenkins Agent Configuration

- Screenshots of Jenkins agent configuration and status

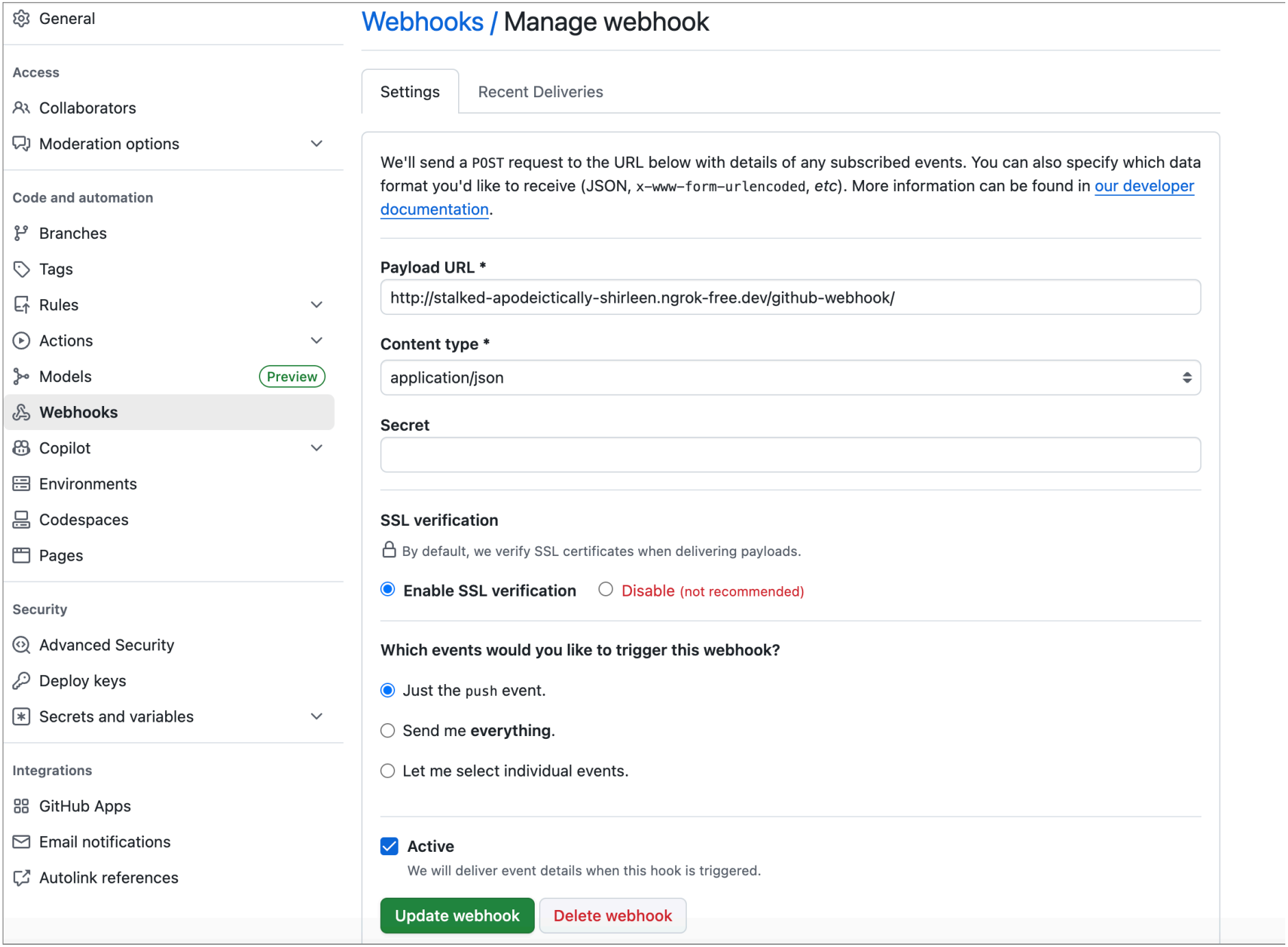

Jenkins / Nodes
Nodes
+ New Node
Configure Monitors
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
S
Name ↓
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
Built-In Node
Mac OS X (aarch64)
In sync
271.55 GiB
1.44 GiB
271.55 GiB
0ms
test-agent
N/A
N/A
N/A
N/A
N/A
Data obtained
46 sec
46 sec
46 sec
45 sec
46 sec
46 sec
Icon: S M L
Legend



- Jenkins agent labels configuration

Question 3: Source Control Integration

- Webhook configuration screenshots from Git repository


General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Models
Preview
Webhooks
Copilot
Environments
Codespaces
Pages
Security
Advanced Security
Deploy keys
Secrets and variables
Integrations
GitHub Apps
Email notifications
Autolink references
Webhooks / Manage webhook
Settings
Recent Deliveries
We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in our developer documentation.
Payload URL *
http://stalked-apodeictically-shirleen.ngrok-free.dev/github-webhook/
Content type *
application/json
Secret
SSL verification
By default, we verify SSL certificates when delivering payloads.
Enable SSL verification
Disable (not recommended)
Which events would you like to trigger this webhook?
Just the push event.
Send me everything.
Let me select individual events.
Active
We will deliver event details when this hook is triggered.
Update webhook
Delete webhook

Jenkins / A4 Pipeline / Configure

Configure

- General
- Triggers
- Pipeline
- Advanced

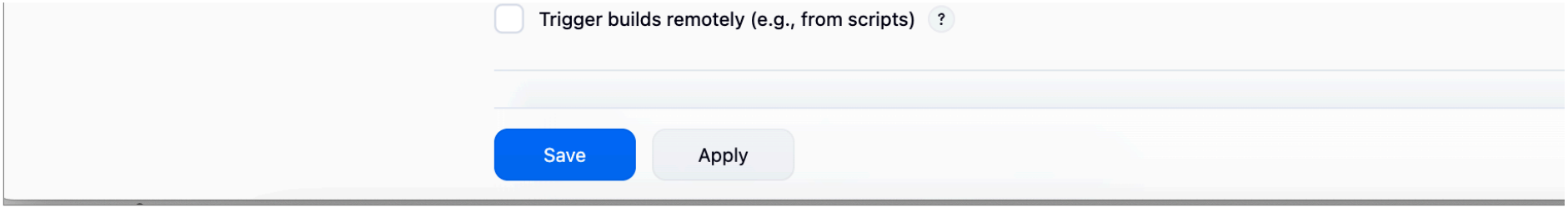
Plain text Preview

- [ ] Discard old builds
- [ ] Do not allow concurrent builds
- [ ] Do not allow the pipeline to resume if the controller restarts
- [x] GitHub project
  - Project url
    - https://github.com/aep5142/a4-devops/
  - Advanced
- [ ] Pipeline speed/durability override
- [ ] Preserve stashes from completed builds
- [ ] This project is parameterized
- [ ] Throttle builds

Triggers

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

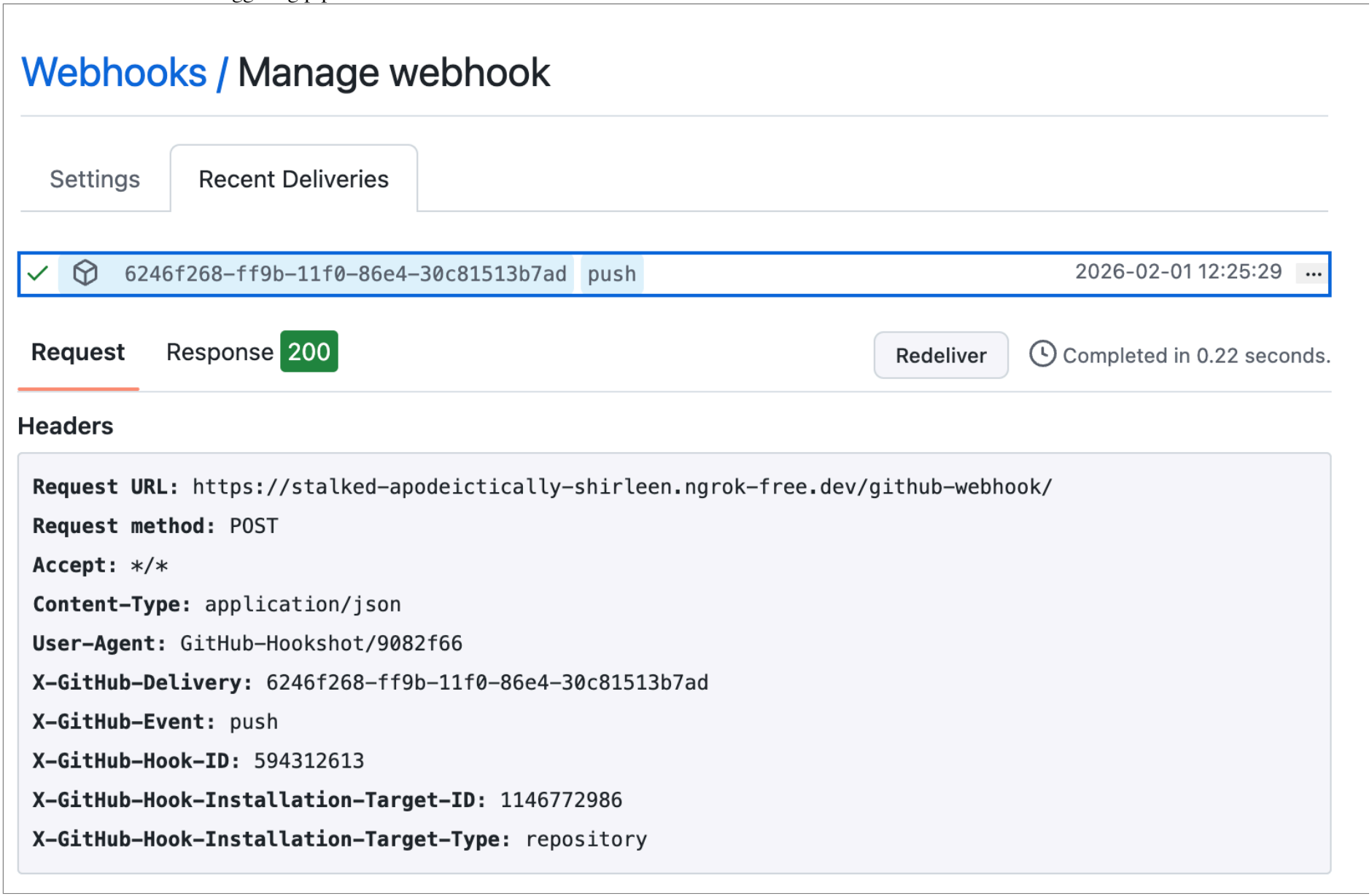
- [ ] Build after other projects are built
- [ ] Build periodically
- [x] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

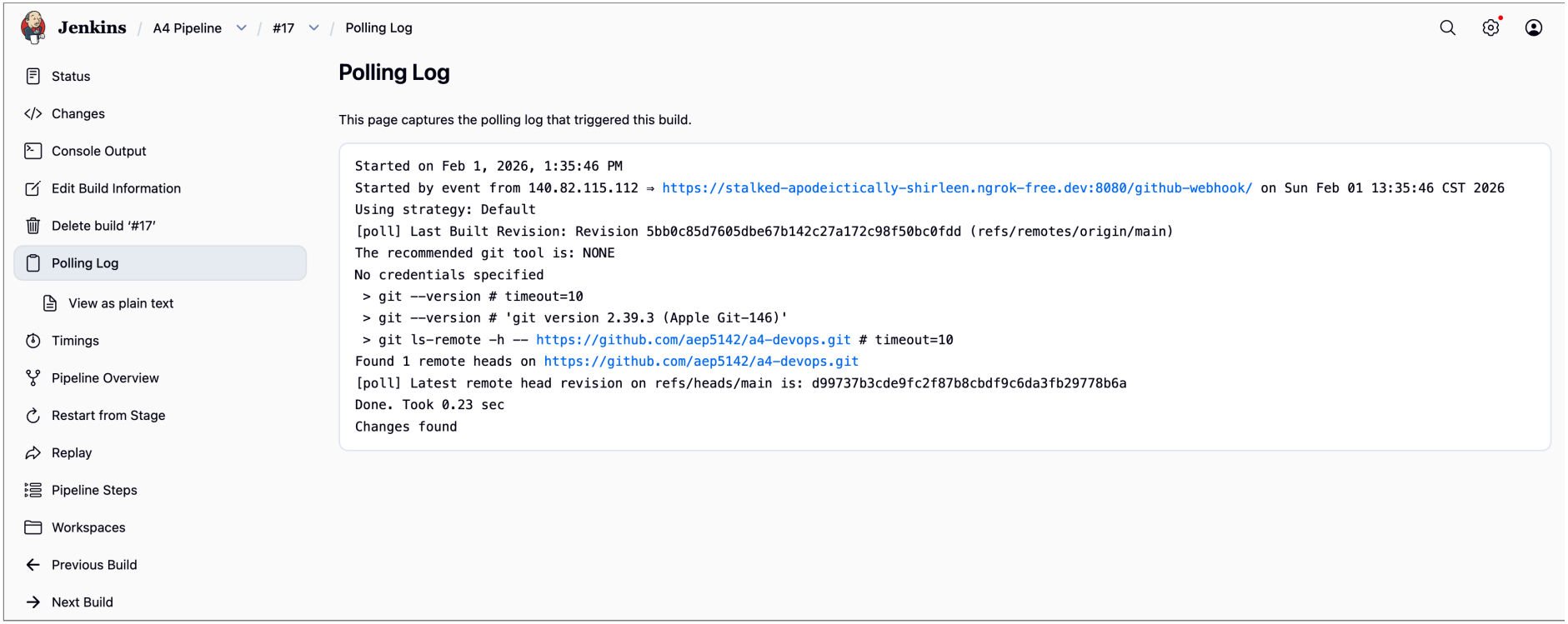


- Jenkinsfile showing branch-specific pipeline logic

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo "Building branch: ${env.BRANCH_NAME}"
            }
        }

        stage('Test') {
            when {
                branch 'main'
            }
            steps {
                echo 'Running tests on main branch'
            }
        }

        stage('Deploy') {
            when {
                branch 'main'
            }
            steps {
                echo 'Deploying application from main branch'
            }
        }

        stage('Feature Checks') {
            when {
                not {
                    branch 'main'
                }
            }
            steps {
                echo 'Running feature branch checks'
            }
        }
    }
}
```

- Screenshot of webhook triggering pipeline runs





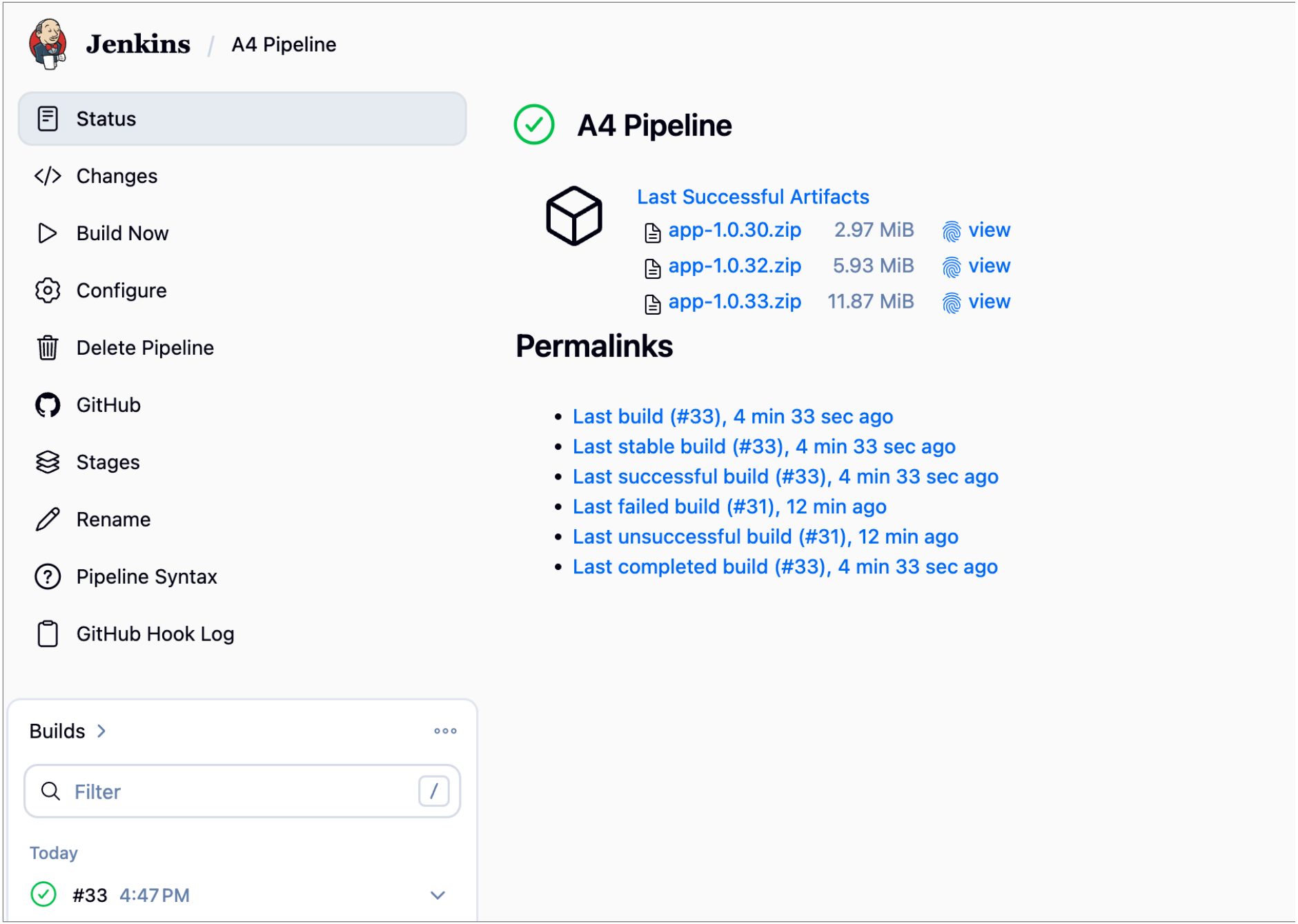
Question 4: Build and Package

- Build logs showing successful artifact creation:

```
Building branch from: origin/main
[Pipeline] echo
Building version 1.0.33
[Pipeline] sh
+ mkdir -p dist
+ zip -r dist/app-1.0.33.zip .
  adding: app/ (stored 0%)
  adding: app/requirements.txt (deflated 21%)
  adding: app/main.py (deflated 70%)
  adding: uv.lock (deflated 66%)
  adding: dist/ (stored 0%)
  adding: dist/app-1.0.30.zip (stored 0%)
  adding: dist/app-1.0.32.zip (stored 0%)
  adding: pyproject.toml (deflated 35%)
  adding: tests/ (stored 0%)
  adding: tests/test_app.py (deflated 72%)
  adding: README.md (stored 0%)
  adding: .gitignore (stored 0%)
  adding: homework_files/ (stored 0%)
  adding: homework_files/screenshots/ (stored 0%)
  adding: homework_files/screenshots/35.png (deflated 17%)
  adding: homework_files/screenshots/21.png (deflated 21%)
  adding: homework_files/screenshots/34.png (deflated 15%)
  adding: homework_files/screenshots/22.png (deflated 30%)
  adding: homework_files/screenshots/23.png (deflated 30%)
  adding: homework_files/screenshots/33.png (deflated 26%)
  adding: homework_files/screenshots/32.png (deflated 18%)
  adding: homework_files/screenshots/31.png (deflated 14%)
  adding: homework_files/index.html (deflated 70%)
  adding: .github/ (stored 0%)
  adding: .github/workflows/ (stored 0%)
  adding: .github/workflows/python-app.yml (deflated 62%)
```

```
Deploying application from main branch
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Feature Checks)
Stage "Feature Checks" skipped due to when conditional
[Pipeline] getContext
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Archive Artifacts)
[Pipeline] archiveArtifacts
Archiving artifacts
Recording fingerprints
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- Screenshots of versioned artifacts in Jenkins:



- Build configuration in Jenkinsfile:

```
    stages {
        stage('Build') {
            steps {
                echo "Building branch from: ${env.BRANCH_PUSH}"

                // Example build artifact
                echo "Building version ${VERSION}"
                sh """
                    mkdir -p dist
                    zip -r dist/${ARTIFACT_NAME} .
                """
            }
        }

        stage('Test') {
            when {
                expression { env.BRANCH_PUSH == 'origin/main' }
            }
            steps {
                echo 'Running tests on main branch!'
            }
        }

        stage('Deploy') {
            when {
                expression { env.BRANCH_PUSH == 'origin/main' }
            }
```

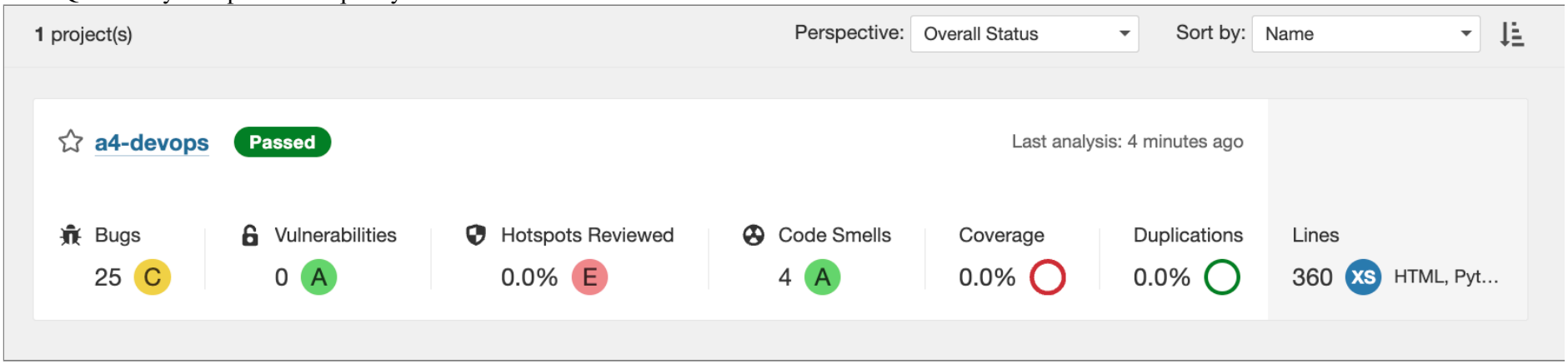
```
        steps {
            echo 'Deploying application from main branch'
        }
    }

    stage('Feature Checks') {
        when {
            expression { env.BRANCH_PUSH != 'origin/main' }
        }
        steps {
            echo 'Running feature branch checks'
        }
    }

    stage('Archive Artifacts') {
        steps {
            archiveArtifacts artifacts: 'dist/*.zip', fingerprint: true
        }
    }
}
```

Question 5: Code Quality Analysis

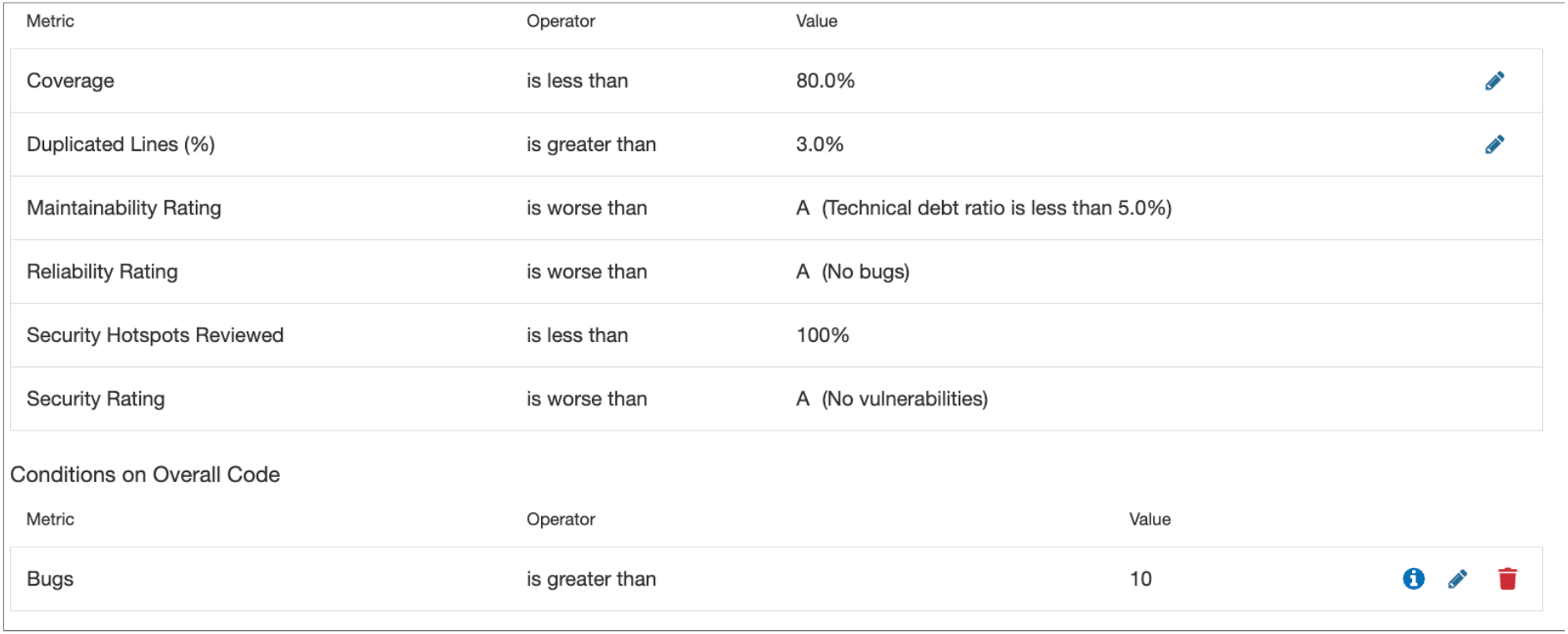
- SonarQube project configuration screenshots

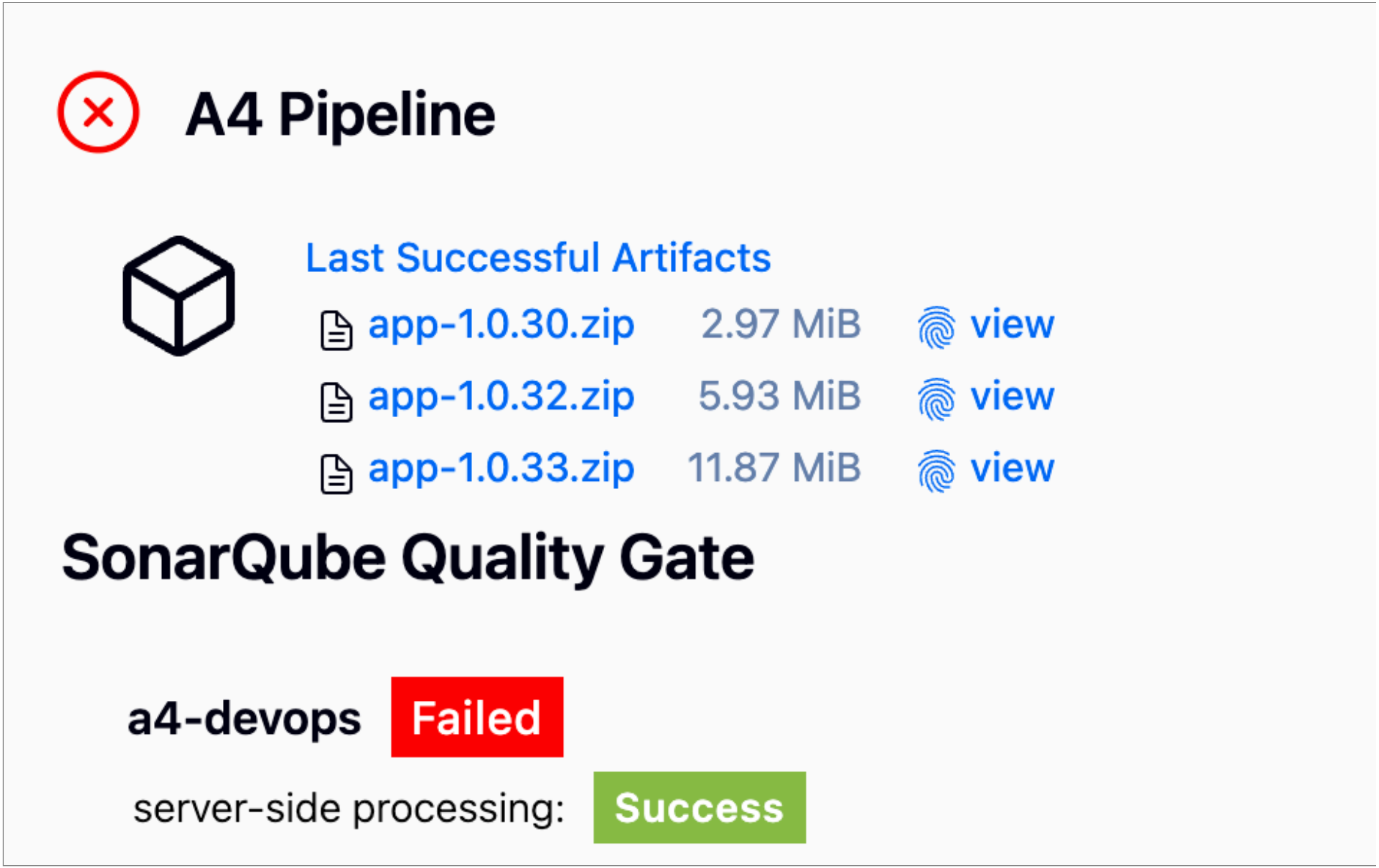
```
        stage('SonarQube Analysis') {
            steps {
                withSonarQubeEnv('SonarQube') {
                    sh """
                    /opt/homebrew/bin/sonar-scanner \
                        -Dsonar.projectKey=a4-devops \
                        -Dsonar.sources=. \
                        -Dsonar.language=py
                    """
                }
            }
        }
    }

        stage('Quality Gate') {
            steps {
                timeout(time: 3, unit: 'MINUTES') {
                    waitForQualityGate abortPipeline: true
                }
            }
        }
```

- SonarQube analysis reports with quality metrics:



- Quality gate configuration and pipeline failure evidence when standards not met
- Here I set only 10 bugs:






```
[Pipeline] waitForQualityGate
Checking status of SonarQube task 'AZwgxpxGGvGMggslpj0d' on server 'SonarQube'
SonarQube task 'AZwgxpxGGvGMggslpj0d' status is 'SUCCESS'
SonarQube task 'AZwgxpxGGvGMggslpj0d' completed. Quality gate is 'ERROR'
```

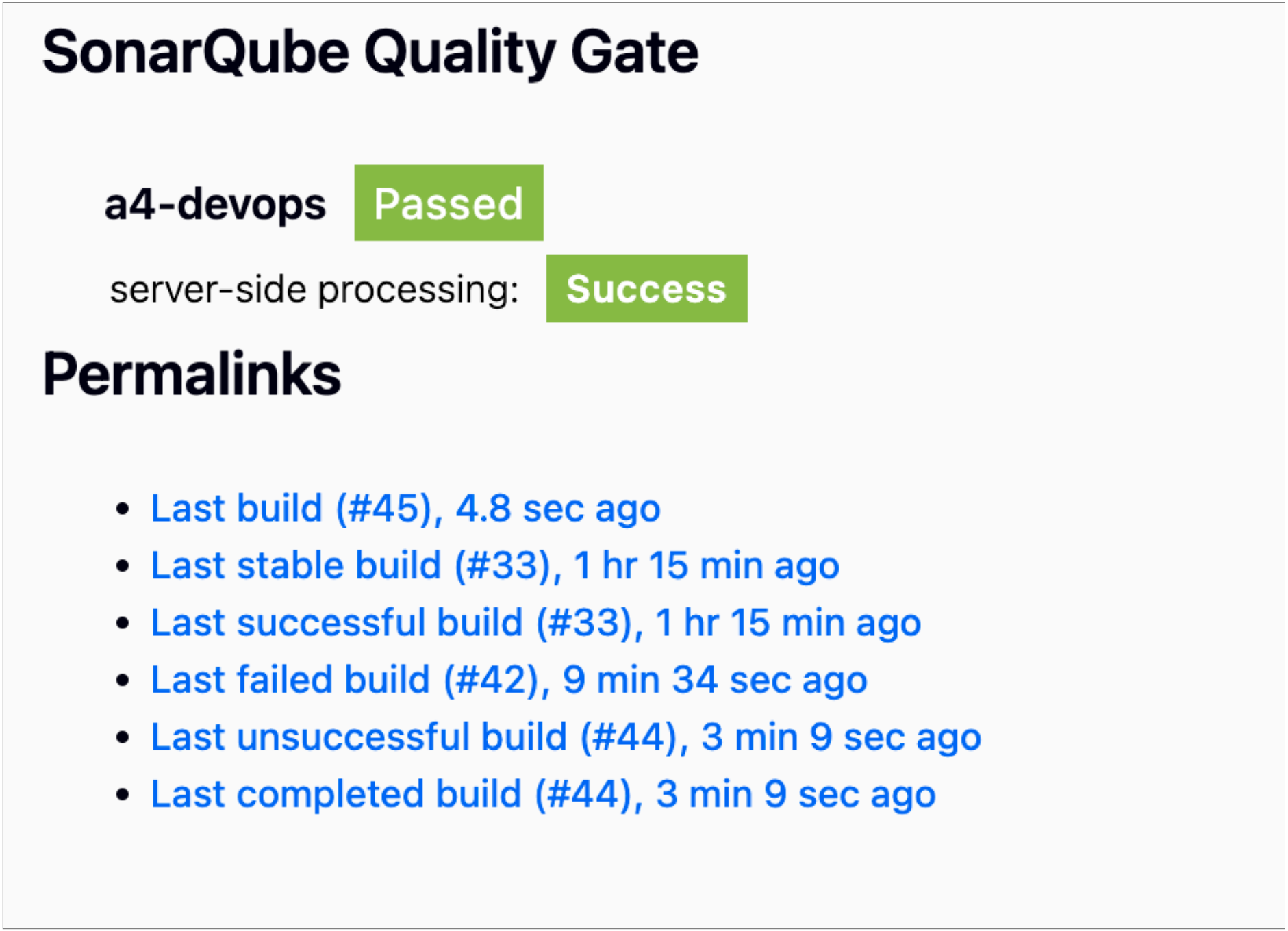
- Here I relaxed all measures to pass the test:

Conditions

Add Condition

Conditions on New Code

Metric	Operator	Value
Coverage	is less than	10.0%
Duplicated Lines (%)	is greater than	30.0%
Maintainability Rating	is worse than	D
Reliability Rating	is worse than	D
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	D



Question 6: Database Management

- Database schema scripts (SQL files or equivalent)

index.html ● Jenkinsfile .env staging_schema.sql U

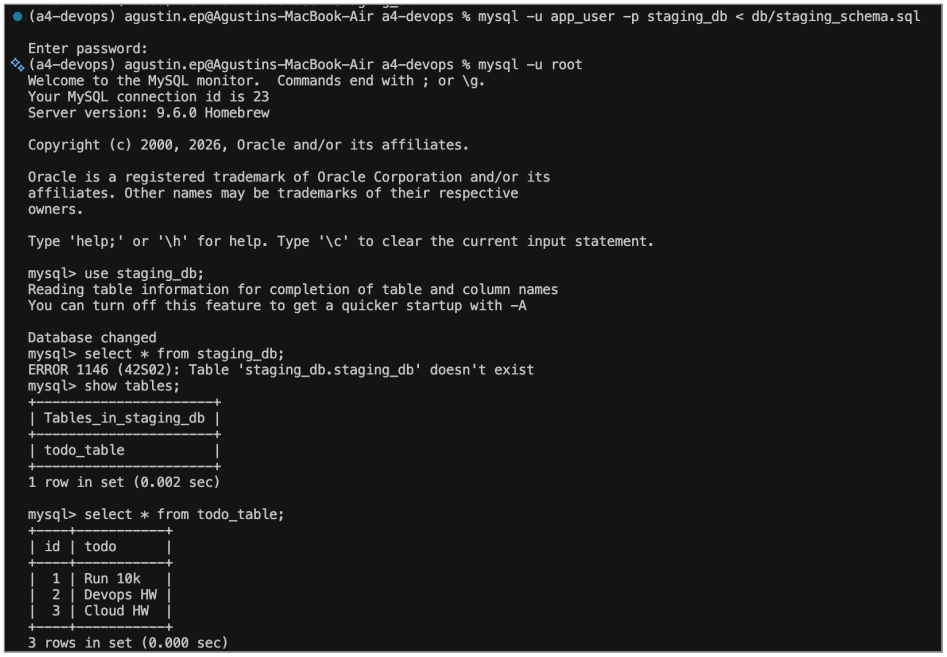
db > staging_schema.sql

```sql
-- Create staging database
CREATE DATABASE IF NOT EXISTS staging_db;

-- Use staging database
USE staging_db;

-- Create todo table
CREATE TABLE IF NOT EXISTS todo_table (
    id INT AUTO_INCREMENT PRIMARY KEY,
    todo VARCHAR(255) NOT NULL
);

-- Insert some test to-do items
INSERT INTO todo_table (todo) VALUES ('Run 10k');
INSERT INTO todo_table (todo) VALUES ('Devops HW');
INSERT INTO todo_table (todo) VALUES ('Cloud HW');
```

- Test data seeding scripts

```python
def test_add_success(monkeypatch):
    # prepare fake cursor that will record executed SQL
    cursor = FakeCursor()

    def fake_connect(**kwargs):
        return FakeConn(cursor)

    monkeypatch.setattr('mysql.connector.connect', fake_connect)

    client = app.test_client()
    resp = client.get('/add?to_add_item=task1')
    assert resp.status_code == 200
    assert b'Added task1 successfully' in resp.data
    assert cursor.executed, "Expected an INSERT to be executed"
    assert 'INSERT INTO todo_table' in cursor.executed[0][0]
```

- Screenshots showing successful database creation and seeding

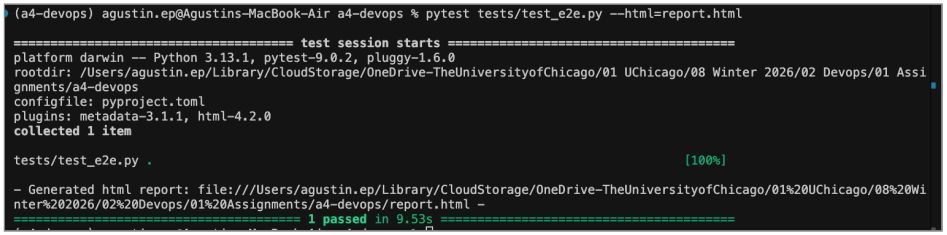


Question 7: End-to-End Testing

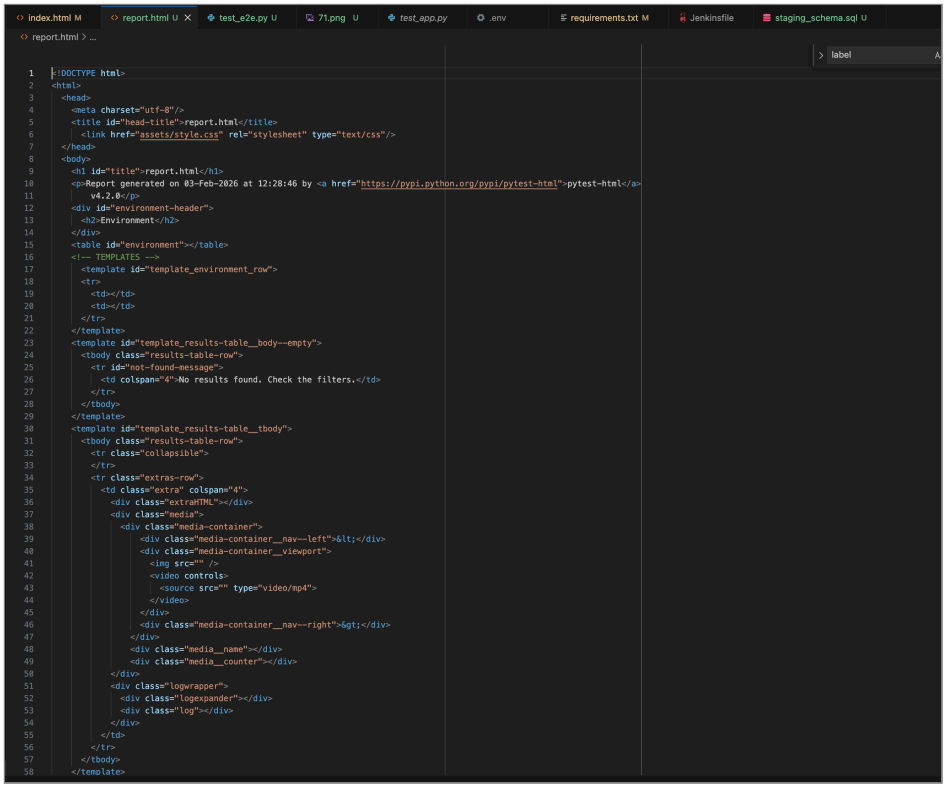
- End-to-end test code/scripts

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
import time
import subprocess

# App URL
BASE_URL = "http://127.0.0.1:5000"

# Start Flask / Uvicorn app
flask_process = subprocess.Popen(
    ["python", "app/main.py", "--host", "127.0.0.1", "--port", "5000"],
    stdout=subprocess.PIPE,
    stderr=subprocess.PIPE,
)

# Wait a few seconds for the app to be ready
time.sleep(5)  # increase if needed

# Initialize Chrome browser
driver = webdriver.Chrome()

def test_e2e():
    try:
        # 1️⃣ Open app
        driver.get(BASE_URL)
        time.sleep(1)  # wait a bit for server

        # 2️⃣ Add a todo
        driver.get(f"{BASE_URL}/add?to_add_item=TestItem")
        time.sleep(0.5)
        assert "Added TestItem successfully" in driver.page_source

        # 3️⃣ View todos
        driver.get(f"{BASE_URL}/view")
        time.sleep(0.5)
        assert "TestItem" in driver.page_source

        # 4️⃣ Delete the todo
        driver.get(f"{BASE_URL}/delete?to_delete_item=TestItem")
        time.sleep(0.5)
        assert "Deleted TestItem successfully" in driver.page_source

        # 5️⃣ Verify deletion
        driver.get(f"{BASE_URL}/view")
        time.sleep(0.5)
        assert "TestItem" not in driver.page_source

        print("End-to-end test passed ✅")

    finally:
        driver.quit()
```

- Test execution screenshots

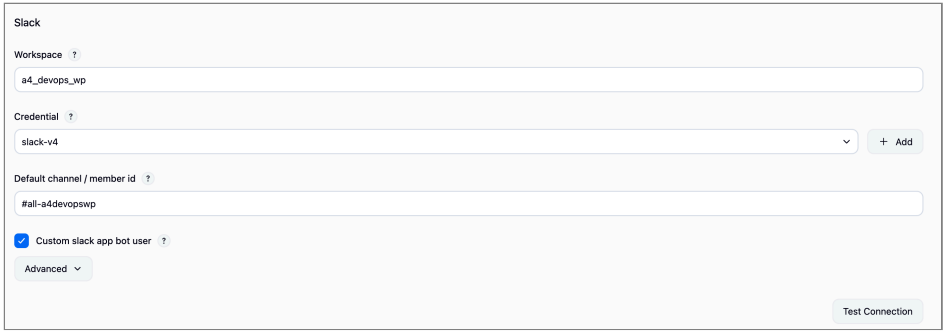


- Generated test reports (HTML, XML, or similar format)

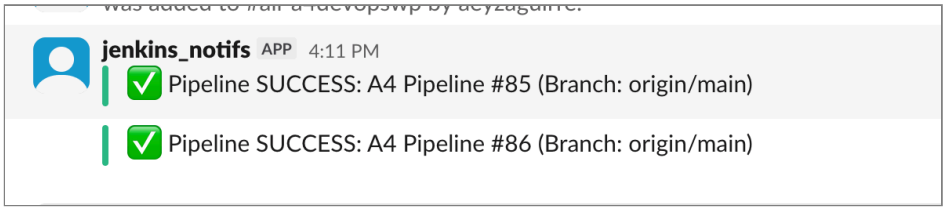


Question 9: Slack Notifications

- Notification configuration in Jenkins



- Screenshots of successful deployment notifications



- Screenshots of failure notifications with error details