

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 <i>mock data</i> to simulate production data	Always with a version control to keep changes. This stage is still very " <i>local</i> " meaning that access here is mostly to everyone but the ones performing changes are developers. Security can be checked with static 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 synthetic 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 standards 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 its 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 should 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 responsible 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 its 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 planning 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

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

+ New Node Configure Monitors

Name ?
test-agent

Description ?
[A4-Devops] This agent will work on testing and deployment tasks

Plain text [Preview](#)

Number of executors ?
1

Remote root directory ?
/home/jenkins

Labels ?
test deploy

Usage ?
Only build jobs with label expressions matching this node

Launch method ?
Launch agent by connecting it to the controller

Availability ?
Keep this agent online as much as possible

[Save](#) [Apply](#)

- Jenkins agent labels configuration

Labels ?

test deploy

Question 3: Source Control Integration

- Webhook configuration screenshots from Git repository

Webhooks / Manage webhook

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Models

Webhooks

Copilot

Environments

Codespaces

Pages

Security

Advanced Security

Deploy keys

Secrets and variables

Integrations

GitHub Apps

Email notifications

Autolink references

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**



Configure

Plain text [Preview](#)

General

Discard old builds [?](#)

Triggers

Do not allow concurrent builds

Pipeline

Do not allow the pipeline to resume if the controller restarts

Advanced

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

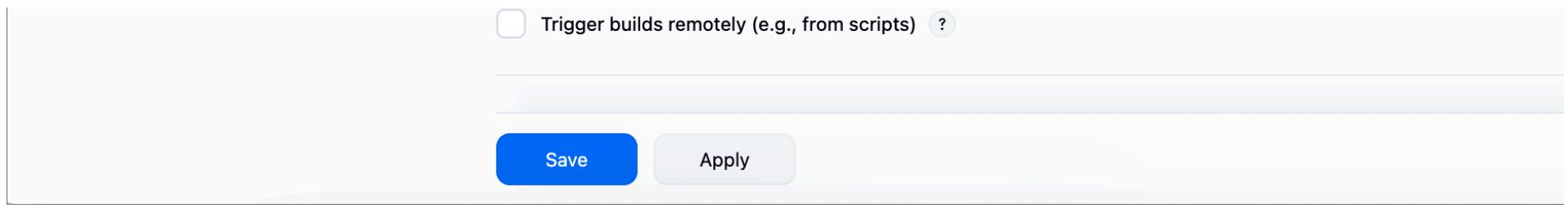
Set up automated actions that start your build based on specific events, like code changes or scheduled times.

Build after other projects are built [?](#)

Build periodically [?](#)

GitHub hook trigger for GITScm polling [?](#)

Poll SCM [?](#)



- Jenkinsfile showing branch-specific pipeline logic

```
1< pipeline {
2     agent any
3
4<     stages {
5<         stage('Build') {
6<             steps {
7                 echo "Building branch: ${env.BRANCH_NAME}"
8             }
9         }
10
11<         stage('Test') {
12             when {
13                 branch 'main'
14             }
15             steps {
16                 echo 'Running tests on main branch'
17             }
18         }
19
20<         stage('Deploy') {
21             when {
22                 branch 'main'
23             }
24             steps {
25                 echo 'Deploying application from main branch'
26             }
27         }
28
29<         stage('Feature Checks') {
30             when {
31                 not {
32                     branch 'main'
33                 }
34             }
35             steps {
36                 echo 'Running feature branch checks'
37             }
38         }
39     }
40 }
```

- Screenshot of webhook triggering pipeline runs

Webhooks / Manage webhook

[Settings](#)[Recent Deliveries](#)

		6246f268-ff9b-11f0-86e4-30c81513b7ad	push	2026-02-01 12:25:29	...
--	--	--------------------------------------	------	---------------------	-----

[Request](#)[Response 200](#)[Redeliver](#)

Completed in 0.22 seconds.

Headers

Request URL: <https://stalked-apodeictically-shirleen.ngrok-free.dev/github-webhook/>

Request method: POST

Accept: */*

Content-Type: application/json

User-Agent: GitHub-Hookshot/9082f66

X-GitHub-Delivery: 6246f268-ff9b-11f0-86e4-30c81513b7ad

X-GitHub-Event: push

X-GitHub-Hook-ID: 594312613

X-GitHub-Hook-Installation-Target-ID: 1146772986

X-GitHub-Hook-Installation-Target-Type: repository

The screenshot shows the Jenkins interface for a pipeline named 'A4 Pipeline'. The current build is '#17'. The 'Polling Log' section is active, showing the log output for triggering this build. The log details the start time (Feb 1, 2026, 1:35:46 PM), the triggering event (an event from IP 140.82.115.112 via a GitHub webhook), and the execution strategy (Default). It also shows the last built revision (5bb0c85d7605dbe67b142c27a172c98f50bc0fdd) and the recommended git tool (NONE). The log ends with a message indicating changes were found.

```
Started on Feb 1, 2026, 1:35:46 PM
Started by event from 140.82.115.112 => https://stalked-apodeictically-shirleen.ngrok-free.dev:8080/github-webhook/ on Sun Feb 01 13:35:46 CST 2026
Using strategy: Default
[poll] Last Built Revision: Revision 5bb0c85d7605dbe67b142c27a172c98f50bc0fdd (refs/remotes/origin/main)
The recommended git tool is: NONE
No credentials specified
> git --version # timeout=10
> git --version # 'git version 2.39.3 (Apple Git-146)'
> git ls-remote -h -- https://github.com/aep5142/a4-devops.git # timeout=10
Found 1 remote heads on https://github.com/aep5142/a4-devops.git
[poll] Latest remote head revision on refs/heads/main is: d99737b3cde9fc2f87b8cbdf9c6da3fb29778b6a
Done. Took 0.23 sec
Changes found
```

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:

The screenshot shows the Jenkins interface for the 'A4 Pipeline'. At the top right, there is a green checkmark icon and the text 'A4 Pipeline'. On the left, a sidebar lists various pipeline management options: Status (highlighted), Changes, Build Now, Configure, Delete Pipeline, GitHub, Stages, Rename, Pipeline Syntax, and GitHub Hook Log. Below the sidebar, the main content area displays 'Last Successful Artifacts' with three entries: 'app-1.0.30.zip' (2.97 MiB), 'app-1.0.32.zip' (5.93 MiB), and 'app-1.0.33.zip' (11.87 MiB), each with a 'view' link. Underneath this is a section titled 'Permalinks' containing links to various build logs: Last build (#33), Last stable build (#33), Last successful build (#33), Last failed build (#31), Last unsuccessful build (#31), and Last completed build (#33). At the bottom left, a 'Builds >' section shows a 'Filter' input field, a date range from 'Today', and a specific build entry '#33 4:47 PM'.

A4 Pipeline

Status

</> Changes

▷ Build Now

⚙ Configure

trash Delete Pipeline

Github GitHub

list Stages

pencil Rename

question Pipeline Syntax

file GitHub Hook Log

Last Successful Artifacts

app-1.0.30.zip 2.97 MiB view

app-1.0.32.zip 5.93 MiB view

app-1.0.33.zip 11.87 MiB view

Permalinks

- [Last build \(#33\), 4 min 33 sec ago](#)
- [Last stable build \(#33\), 4 min 33 sec ago](#)
- [Last successful build \(#33\), 4 min 33 sec ago](#)
- [Last failed build \(#31\), 12 min ago](#)
- [Last unsuccessful build \(#31\), 12 min ago](#)
- [Last completed build \(#33\), 4 min 33 sec ago](#)

Builds >

Filter /

Today

#33 4:47 PM

- 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' }
        }
    }
}
```

```
        }
    }

    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
        }
    }
}
```

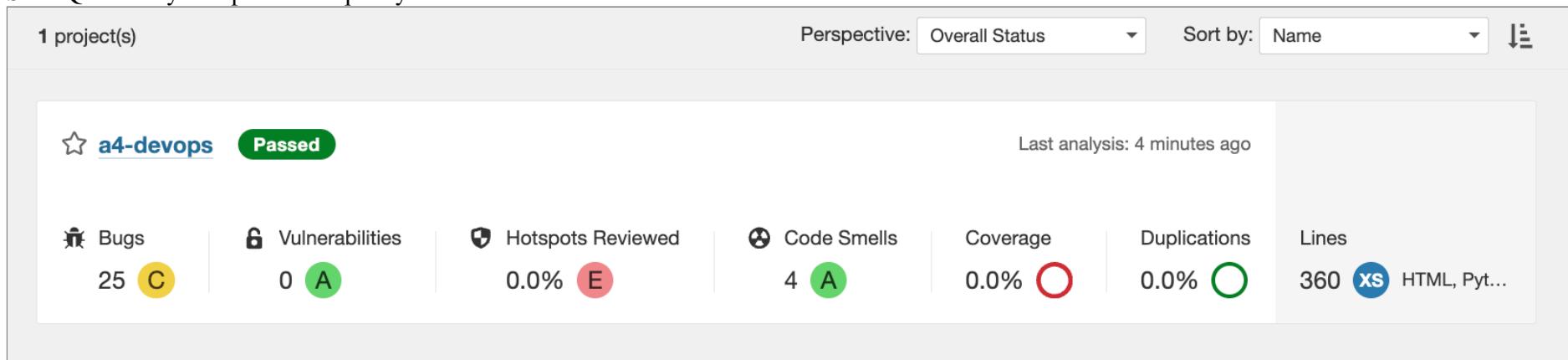
Question 5: Code Quality Analysis

- 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:

Metric	Operator	Value	
Coverage	is less than	80.0%	
Duplicated Lines (%)	is greater than	3.0%	
Maintainability Rating	is worse than	A (Technical debt ratio is less than 5.0%)	
Reliability Rating	is worse than	A (No bugs)	
Security Hotspots Reviewed	is less than	100%	
Security Rating	is worse than	A (No vulnerabilities)	

Conditions on Overall Code

Metric	Operator	Value	
Bugs	is greater than	10	



A4 Pipeline



Last Successful Artifacts

	app-1.0.30.zip	2.97 MiB	 view
	app-1.0.32.zip	5.93 MiB	 view
	app-1.0.33.zip	11.87 MiB	 view

SonarQube Quality Gate

a4-devops Failed

server-side processing: Success

[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:

[Add Condition](#)

Conditions

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	 

SonarQube Quality Gate

a4-devops Passed

server-side processing: Success

Permalinks

- [Last build \(#45\), 4.8 sec ago](#)
- [Last stable build \(#33\), 1 hr 15 min ago](#)
- [Last successful build \(#33\), 1 hr 15 min ago](#)
- [Last failed build \(#42\), 9 min 34 sec ago](#)
- [Last unsuccessful build \(#44\), 3 min 9 sec ago](#)
- [Last completed build \(#44\), 3 min 9 sec ago](#)

Question 6: Database Management

- Database schema scripts (SQL files or equivalent)

The screenshot shows a terminal window with several tabs at the top: 'index.html', 'Jenkinsfile', '.env', and 'staging_schema.sql'. The 'staging_schema.sql' tab is active, displaying the following SQL code:

```

db > # staging_schema.sql
1 -- Create staging database
2 CREATE DATABASE IF NOT EXISTS staging_db;
3
4 -- Use staging database
5 USE staging_db;
6
7 -- Create todo table
8 CREATE TABLE IF NOT EXISTS todo_table (
9     id INT AUTO_INCREMENT PRIMARY KEY,
10    todo VARCHAR(255) NOT NULL
11 );
12
13 -- Insert some test to-do items
14 INSERT INTO todo_table (todo) VALUES ('Run 10k');
15 INSERT INTO todo_table (todo) VALUES ('Devops HW');
16 INSERT INTO todo_table (todo) VALUES ('Cloud HW');

```

- Test data seeding scripts

```

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

```
● (a4-devops) agustin.ep@Agustins-MacBook-Air a4-devops % mysql -u app_user -p staging_db < db/staging_schema.sql
Enter password:
❖ (a4-devops) agustin.ep@Agustins-MacBook-Air a4-devops % mysql -u root
Welcome to MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 9.6.0 Homebrew

Copyright (c) 2000, 2026, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use staging_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from staging_db;
ERROR 1146 (42S02): Table 'staging_db.staging_db' doesn't exist
mysql> show tables;
+-----+
| Tables_in_staging_db |
+-----+
| todo_table           |
+-----+
1 row in set (0.002 sec)

mysql> select * from todo_table;
+----+-----+
| id | todo      |
+----+-----+
|  1 | Run 10k   |
|  2 | Devops HW |
|  3 | Cloud HW  |
+----+-----+
3 rows in set (0.000 sec)
```

Question 7: End-to-End Testing

- End-to-end test code/scripts

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

        # 2 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

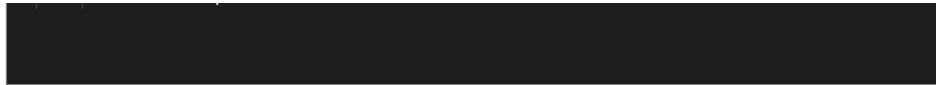
        # 3 View todos
        driver.get(f"{BASE_URL}/view")
        time.sleep(0.5)
        assert "TestItem" in driver.page_source

        # 4 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

        # 5 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

```
(a4-devops) agustin.ep@Agustins-MacBook-Air a4-devops % pytest tests/test_e2e.py --html=report.html
=====
 test session starts =====
platform darwin -- Python 3.13.1, pytest-9.0.2, pluggy-1.6.0
rootdir: /Users/agustin.ep/Library/CloudStorage/OneDrive-TheUniversityofChicago/01 UChicago/08 Winter 2026/02 Devops/01 Assi
gments/a4-devops
configfile: pyproject.toml
plugins: metadata-3.1.1, html-4.2.0
collected 1 item

tests/test_e2e.py . [100%]

- Generated html report: file:///Users/agustin.ep/Library/CloudStorage/OneDrive-TheUniversityofChicago/01%20UChicago/08%20Wi
nter%202026/02%20Devops/01%20Assignments/a4-devops/report.html -
=====
 1 passed in 9.53s =====
```

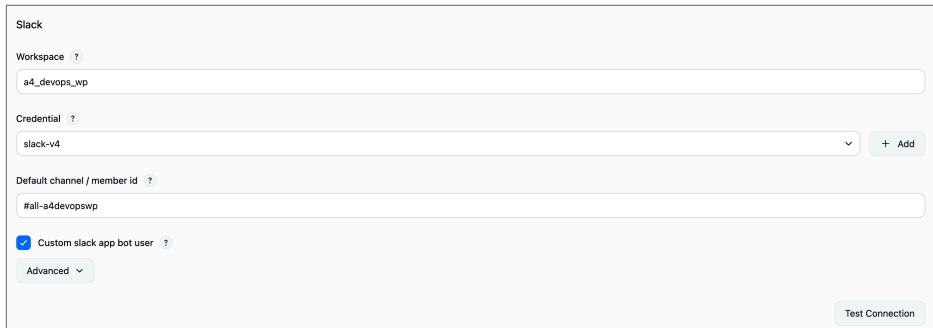
- Generated test reports (HTML, XML, or similar format)

```

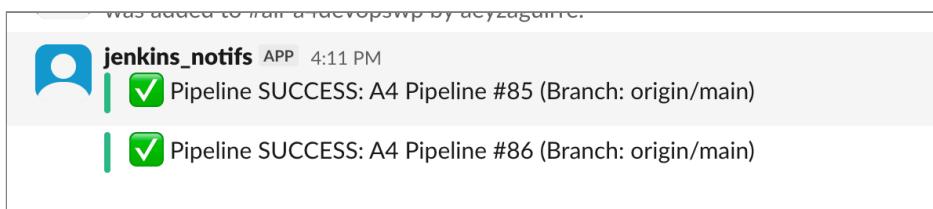
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8"/>
5     <title id="head-title">report.html</title>
6     <link href="assets/style.css" rel="stylesheet" type="text/css"/>
7   </head>
8   <body>
9     <h1 id="title">report.html</h1>
10    <p>Report generated on 03-Feb-2026 at 12:28:46 by <a href="https://pypi.python.org/pypi/pytest-html">pytest-html</a>
11      v4.2.0</p>
12      <div><h2>Environment</h2>
13        <table id="environment"></table>
14      </div>
15      <table id="environments"></table>
16      <!-- TEMPLATES -->
17      <template id="template_environment_row">
18        <tr>
19          <td></td>
20          <td></td>
21        </tr>
22      </template>
23      <template id="template_results-table_body--empty">
24        <tbody class="results-table-row">
25          <tr id="not-found-message">
26            <td colspan="4">No results found. Check the filters.</td>
27          </tr>
28        </tbody>
29      </template>
30      <template id="template_results-table__tbody">
31        <tbody class="results-table-row">
32          <tr class="collapsible">
33            <td></td>
34            <td class="extra-row">
35              <td class="extra" colspan="4">
36                <div class="extraHTML"></div>
37                <div class="media">
38                  <div class="media-container">
39                    <div class="media-container__nav--left">&lt;&lt;&gt;</div>
40                    <div class="media-container__viewport">
41                      <img src="" />
42                      <video controls>
43                        <source src="" type="video/mp4">
44                      </video>
45                    </div>
46                    <div class="media-container__nav--right">&gt;&gt;&lt;&gt;</div>
47                  </div>
48                  <div class="media__name"></div>
49                  <div class="media__counter"></div>
50                </div>
51                <div class="logWrapper">
52                  <div class="logExpander"></div>
53                  <div class="log"></div>
54                </div>
55              </td>
56            </tr>
57          </tbody>
58        </template>
```

Question 9: Slack Notifications

- Notification configuration in Jenkins



- Screenshots of successful deployment notifications



- Screenshots of failure notifications with error details

