

VCS questions with answers

Batch 1

Set 1:

1. Jira Setup:

Your team is starting a new project and needs a Jira Cloud instance to track progress. How would you set up a Jira Cloud instance and navigate to the project board?

Answer:

To set up a Jira Cloud instance and view your project board:

Steps:

1. Go to <https://www.atlassian.com/software/jira>.
 2. Sign in or create an Atlassian account.
 3. Select **Jira Software** and choose a **Scrum** or **Kanban** template.
 4. Enter a project name and key.
 5. Click **Create project**.
 6. Once the project is created, go to the sidebar and click **Board** to view it.
-

2. Basic Git Operations:

You've updated a project file locally and need to push the changes to GitHub. What Git commands would you use to stage, commit, and push the changes?

Answer:

Use these Git commands on Ubuntu:

Steps:

1. Open terminal in the project folder.
2. Stage changes:
git add .

3. Commit changes with a message:
git commit -m "Updated project file"
 4. Push to GitHub:
git push origin main
-

3. JQL Search:

Your manager asks you to find all Jira issues assigned to you with the status "In Progress." What JQL query would you use to find this information?

Answer:

JQL Query:

assignee = currentUser() AND status = "In Progress"

Set 2:

1. GitHub Repository:

You're collaborating with a colleague on a new project. How would you create a GitHub repository and share it with your teammate?

Answer:

You can create and share a repository as follows:

Steps:

1. Go to <https://github.com> and log in.
 2. Click **New Repository**.
 3. Add a name, description, and select visibility.
 4. Click **Create repository**.
 5. After creation, click **Settings > Collaborators**.
 6. Add your teammate's GitHub username and click **Add**.
-

2. Jira Scrum Project:

Your team is following the Scrum methodology. How would you create and configure a new Scrum project in Jira?

Answer:

To create a Scrum project:

Steps:

1. Log in to Jira Cloud.
 2. Click **Projects > Create Project**.
 3. Select the **Scrum** template.
 4. Enter project details and click **Create**.
 5. Configure boards and sprints from the project's sidebar.
-

3. Git Branching:

You want to create a new feature without affecting the main code. How would you create a new branch, switch to it, and merge it back once the feature is complete?

Answer:

Use Git branching as follows:

Steps:

1. Create new branch:
git branch feature-branch
 2. Switch to it:
git checkout feature-branch
 3. After coding, switch back to main:
git checkout main
 4. Merge the branch:
git merge feature-branch
-

Set 3:

1. Git Installation:

You've set up a new laptop and need to configure Git with your GitHub account. What steps would you take to install Git and link it with your GitHub username and email?

Answer:

Use Ubuntu terminal:

Steps:

1. Install Git:
`sudo apt update && sudo apt install git -y`
 2. Set username:
`git config --global user.name "Your Name"`
 3. Set email:
`git config --global user.email "you@example.com"`
-

2. Jira Issue Creation:

You've identified a bug in the project. How would you create an issue in Jira, classify it under the appropriate epic, and assign it to a developer?

Answer:

Create and assign an issue in Jira:

Steps:

1. Go to the project board.
 2. Click **Create** in the top menu.
 3. Fill in the summary, issue type as **Bug**, and link it to the relevant epic.
 4. Set the priority and assign the developer.
 5. Click **Create**.
-

3. Jira Filter Creation:

You want to track only high-priority issues in your project. How would you create a JQL filter and save it for future use?

Answer:

Use JQL filter like this:

Steps:

1. Go to **Filters > Advanced Issue Search**.
 2. Enter query:
priority = High
 3. Click **Save As**, give it a name, and save the filter.
-

I'll continue with **Set 4 to Set 8** next. Shall I go on?

4o

You said:

yes

ChatGPT said:

Continuing with the remaining sets from **Batch 1**, here are the answers for:

Set 4:

1. Local Git Initialization:

You've just created a new project folder on your local machine. How would you initialize a Git repository and link it to your GitHub repository?

Answer:

Steps:

1. Open terminal and navigate to the project folder:
cd your-project-folder
2. Initialize Git locally:
git init
3. Add files:
git add .
4. Commit:
git commit -m "Initial commit"
5. Add remote GitHub repo:
git remote add origin <https://github.com/yourusername/your-repo.git>

6. Push to GitHub:
git push -u origin main
-

2. Git Push and Pull:

How would you pull updates into your local repository without losing your work?

Answer:

Steps:

1. Save any current changes using:
git add . && git commit -m "Saving local changes"
 2. Pull updates from GitHub:
git pull origin main
 3. If conflicts occur, resolve them manually and re-commit:
git commit -m "Resolved merge conflicts"
-

3. Jira Dashboard:

You need to create a dashboard in Jira showing open issues, active sprints, and completed tasks.

Answer:

Steps:

1. Go to Jira and click **Dashboards > Create dashboard**.
 2. Name the dashboard and set visibility.
 3. Add gadgets:
 - **Filter Results** for open issues.
 - **Sprint Burndown** for active sprint.
 - **Two Dimensional Filter Statistics** for completed tasks.
 4. Save and arrange widgets as needed.
-

Set 5:

1. Jira Navigation:

How would you find the sprint backlog and the active sprint board?

Answer:

Steps:

1. Open the Jira project.
 2. In the sidebar, click on **Backlog** to view sprint backlog.
 3. Click **Active sprints** to see the current sprint board.
-

2. Git Commit:

What Git commands would you use to add the file and commit the changes with a descriptive message?

Answer:

Steps:

1. Add the modified file:
git add filename.ext
 2. Commit with a message:
git commit -m "Updated XYZ functionality"
-

3. Git Merge:

What steps would you follow to merge a feature branch into the main branch smoothly?

Answer:

Steps:

1. Switch to main branch:
git checkout main
2. Pull latest changes:
git pull origin main

3. Merge feature branch:
git merge feature-branch
 4. Push merged changes:
git push origin main
-

Set 6:

1. GitHub Setup:

How would you create a GitHub repository with a README.md and .gitignore for .java files?

Answer:

Steps:

1. Go to GitHub and click **New repository**.
 2. Enter repo name and description.
 3. Check the boxes for:
 - **Add README.md**
 - **.gitignore > Java**
 4. Click **Create repository**.
-

2. Jira Epic Creation:

How would you create an epic in Jira and link relevant stories to it?

Answer:

Steps:

1. Click **Create** > Select **Epic** as issue type.
 2. Enter details and click **Create**.
 3. To link stories:
 - Open story issue
 - In the **Epic Link** field, choose the created epic.
-

3. JQL Search:

What JQL query would you use to find all issues in the current sprint that are still open?

Answer:

JQL Query:

sprint in openSprints() AND resolution = Unresolved

Set 7:

1. Git Commit:

How would you add a file and commit the changes with a message?

Answer:

Steps:

1. Stage the file:
git add filename.ext
 2. Commit it:
git commit -m "Added new changes to XYZ"
-

2. Git Pull:

How would you pull changes from main without conflicts?

Answer:

Steps:

1. Make sure your branch is clean:
git status
 2. Pull changes:
git pull origin main
 3. If conflicts appear, resolve and commit again.
-

3. Jira Board Configuration:

How would you configure a Kanban board in Jira and customize its columns?

Answer:

Steps:

1. Go to **Project settings > Boards > Create Board**.
 2. Choose **Kanban** template.
 3. Name the board and select a project.
 4. Click **Board settings > Columns** to add or rename columns.
-

Set 8:

1. JQL Search:

Find all Jira issues assigned to you with the status "In Progress".

Answer:

JQL Query:

assignee = currentUser() AND status = "In Progress"

2. Jira Story Creation:

How would you create a Jira story with appropriate details and assign it to the development team?

Answer:

Steps:

1. Click **Create** in Jira.
 2. Choose issue type: **Story**.
 3. Add title, description, acceptance criteria, priority.
 4. Assign to a developer or team.
 5. Click **Create**.
-

3. Git Branching:

How would you create a separate branch for a hotfix and merge it into main without affecting the other feature branch?

Answer:

Steps:

1. Create hotfix branch:
git checkout -b hotfix-branch
2. Make changes, then commit:
git add .
git commit -m "Hotfix implemented"
3. Switch to main and merge:
git checkout main
git merge hotfix-branch

Set 9:

1. Jira Cloud Instance:

How would you create user accounts and assign project roles within your Jira Cloud instance?

Answer:

Steps:

1. Log in to Jira Cloud as an admin.
2. Go to **Settings > User Management**.
3. Click **Invite users**, enter emails, and select product access.
4. Open the project, go to **Project Settings > People**.
5. Add users and assign roles like **Developer**, **Admin**, etc.

2. Git Commit and Push:

How would you stage all the changes, commit them with a message, and push them to GitHub?

Answer:

Steps:

1. Open terminal and navigate to your project.
2. Stage all files:
git add .
3. Commit with message:
git commit -m "Updated project files"
4. Push to GitHub:
git push origin main

3. JQL Search:

Find issues with the label "Bug" created in the last 7 days.

Answer:

JQL Query:

labels = Bug AND created >= -7d

Set 10:

1. GitHub Repository Setup:

How would you create a repository, add a license file, and enable branch protection rules?

Answer:

Steps:

1. Go to <https://github.com> > **New Repository**.
 2. Fill repo details, check **Add a license**.
 3. After creation, go to **Settings > Branches**.
 4. Under **Branch protection rules**, click **Add rule**.
 5. Add rule for main and enable “Require pull request reviews”.
-

2. Jira Sprint Creation:

How would you create a sprint and move backlog items into it?

Answer:

Steps:

1. Go to the Jira project’s **Backlog** view.
 2. Click **Create Sprint** at the top.
 3. Drag backlog issues into the sprint area.
 4. Click **Start Sprint**, set start and end dates.
-

3. Git Branch Creation:

How would you create a new branch, switch to it, and push it to GitHub?

Answer:

Steps:

1. Create branch:
git checkout -b feature-branch
 2. Push to GitHub:
git push -u origin feature-branch
-

Set 11:

1. Git Installation and Configuration:

How would you install Git and configure it with your GitHub credentials?

Answer:

Steps:

1. Install Git:
sudo apt update && sudo apt install git -y
 2. Set username:
git config --global user.name "Your Name"
 3. Set email:
git config --global user.email "you@example.com"
-

2. Jira Issue Creation:

How would you create a Jira issue, set the priority to "High," and assign it to a developer?

Answer:

Steps:

1. Click **Create** in Jira.
2. Select project and issue type (e.g., Bug or Task).
3. Enter summary and description.
4. Set priority to **High**.

5. Assign to a developer.
6. Click **Create**.

3. Jira Dashboard Setup:

How would you create and configure a dashboard showing sprint progress and issue breakdown?

Answer:

Steps:

1. Go to **Dashboards > Create Dashboard**.
2. Name it, set permissions, and click **Create**.
3. Click **Add Gadget**.
4. Add:
 - **Sprint Health**
 - **Issue Statistics**
 - **Two Dimensional Filter Stats**
5. Configure each gadget using saved filters.

Set 12:

1. Local Git Initialization:

How would you initialize it as a Git repository and link it to GitHub?

Answer:

Steps:

1. Navigate to folder:
cd project-folder
2. Initialize Git:
git init

3. Add files:
git add .
 4. Commit:
git commit -m "Initial commit"
 5. Add remote:
git remote add origin <https://github.com/your/repo.git>
 6. Push:
git push -u origin main
-

2. Git Pull and Merge:

How would you pull changes and handle merge conflicts?

Answer:

Steps:

1. Pull changes:
git pull origin main
 2. If merge conflicts occur, open files and resolve them.
 3. After resolving, stage files:
git add conflicted-file
 4. Commit:
git commit -m "Resolved merge conflicts"
-

3. Jira Filter Creation:

How would you create a JQL filter and share it with the team?

Answer:

Steps:

1. Go to **Filters > Advanced Issue Search**.
2. Write query:
assignee in (membersOf("team-name")) AND resolution = Unresolved

3. Click **Save As**, name the filter.
 4. Go to **Manage filters > Share** to share with team.
-

Set 13:

1. Jira Project Setup:

How would you set up a new Scrum project in Jira and configure permissions?

Answer:

Steps:

1. Go to **Projects > Create Project**.
 2. Select **Scrum** and click **Use template**.
 3. Fill in project name and click **Create**.
 4. Go to **Project settings > People**, add client emails and assign roles.
-

2. Git Add and Commit:

How would you stage only specific files and commit?

Answer:

Steps:

1. Stage selected files:
git add file1.js file2.css
 2. Commit:
git commit -m "Updated styles and scripts"
 3. Push:
git push origin main
-

3. Git Merge:

How to merge a branch into main without fast-forwarding?

Answer:

Steps:

1. Switch to main:
git checkout main
 2. Pull latest:
git pull origin main
 3. Merge with no fast-forward:
git merge --no-ff feature-branch
 4. Push changes:
git push origin main
-

Set 14:

1. GitHub Repository Collaboration:

How would you invite an external developer and set access?

Answer:

Steps:

1. Go to repo > **Settings** > **Collaborators**.
 2. Click **Invite a collaborator**.
 3. Enter GitHub username and click **Add**.
 4. Set access to **Read**, **Write**, or **Admin** as needed.
-

2. Jira Epic and Story Linking:

How would you create an epic and link stories to it?

Answer:

Steps:

1. Create an Epic via **Create** > **Issue Type** > **Epic**.
2. Add title and description, then click **Create**.
3. Open a story issue.
4. In **Epic Link**, select your epic.

3. Git Branch Deletion:

How would you safely delete a branch locally and on GitHub?

Answer:

Steps:

1. Delete local branch:
git branch -d feature-branch
2. Delete remote branch:
git push origin --delete feature-branch

Set 15:

1. Git Configuration Update:

How would you update Git email address?

Answer:

Steps:

1. Change email:
git config --global user.email "newemail@example.com"

2. Jira Sprint Management:

How would you close a sprint and move unfinished tasks?

Answer:

Steps:

1. Go to **Active sprints**.
2. Click **Complete Sprint**.
3. Jira will prompt to move incomplete issues to backlog or next sprint.

3. JQL Advanced Search:

Find all issues assigned to you, excluding those marked "Done".

Answer:

JQL Query:

assignee = currentUser() AND status != Done

Set 16:

1. Jira Filter Creation:

Track all open issues assigned to your team.

Answer:

JQL Query:

assignee in (membersOf("team-name")) AND resolution = Unresolved

Steps:

1. Go to **Filters > Advanced Search**, enter the query.
 2. Click **Save As** and share with team.
-

2. Jira Kanban Board Configuration:

How to configure a Kanban board and customize its columns?

Answer:

Steps:

1. Go to **Boards > Create board > Kanban**.
 2. Choose the project and name the board.
 3. Go to **Board settings > Columns** to rename/add/remove columns.
-

3. Jira Issue Creation:

How would you create a critical issue with high priority?

Answer:

Steps:

1. Click **Create** in Jira.

2. Select the appropriate project.
3. Fill in title, description, and set priority to **High**.
4. Assign to a developer and click **Create**.

CIA 2 Batch 1

Set 1:

Q1. Git Scenario – Project Initialization

You're starting a new project called InvoiceApp. Describe how to:

Answer:

Steps:

1. Open terminal and navigate to project folder: **cd InvoiceApp**
2. Initialize Git:
git init
3. Create GitHub repo manually on GitHub.
4. Add remote: **git remote add origin**
<https://github.com/yourusername/InvoiceApp.git>
5. Stage all files: **git add .**
6. Commit: **git commit -m "Initial commit"**
7. Push: **git push -u origin main**

Q2. JQL Advanced Search – Created, Due, and Resolution Filters

a. Find all issues created in the last 10 days that are still unresolved

JQL:

created >= -10d AND resolution = Unresolved

b. Show all issues that are due in the next 3 days

JQL:

due <= 3d

c. Find all bugs that were resolved in the last 5 days

JQL:

issuetype = Bug AND resolved >= -5d

Explanation:

- **resolution = EMPTY** checks if no resolution is set (i.e., unresolved).
 - **status != Done** may include resolved issues still in progress stages (like “Review”).
-

Set 2:

Q1. GitHub Scenario – Team Collaboration

Steps:

1. Go to GitHub > **New Repository**.
 2. Add a name and check **Add README.md**.
 3. Create repository.
 4. Go to **Settings > Collaborators** > add team members.
 5. Enable branch protection:
 - **Settings > Branches > Add rule**
 - Protect **main**, require pull request reviews.
-

Q2. JQL Search – Transitions, Assignee, Due Dates

- a. Status changed from "To Do" to "In Progress" in the last 7 days

JQL:

status CHANGED FROM "To Do" TO "In Progress" AFTER -7d

- b. All issues assigned to you that are overdue

JQL:

assignee = currentUser() AND due < now()

- c. Tasks that transitioned to “Done” in the last 2 days

JQL:

status CHANGED TO Done AFTER -2d

Explanation:

- **CHANGED** tracks transitions between statuses over time.
-

Set 3:

Q1. Git Scenario – Branching & PR

Steps:

1. Create a new branch: **git checkout -b search-feature**
 2. Make changes and stage: **git add .**
 3. Commit: **git commit -m "Added search feature"**
 4. Push: **git push -u origin search-feature**
 5. Go to GitHub > Pull Requests > New PR.
 6. Assign reviewer and submit.
 7. After approval, click **Merge**.
-

Q2. JQL – Created/Updated/Transitioned Filter Queries

- a. Issues created between March 1 and March 10, 2025

JQL:

created >= "2025-03-01" AND created <= "2025-03-10"

- b. Issues updated within the last 3 days

JQL:

updated >= -3d

- c. Issues transitioned from “In Progress” to “Testing” after April 1, 2025

JQL:

status CHANGED FROM "In Progress" TO "Testing" AFTER "2025-04-01"

Explanation:

- Use **updated >= -3d** for recent edits, and **created >= -3d** for newly added issues.
-

Set 4:

Q1. Git Scenario – Resolve Merge Conflict

Steps:

1. Pull remote updates: **git pull origin main**
 2. If there's a conflict, Git marks it in files (<<<<, =====, >>>>).
 3. Manually edit the conflicting file to resolve.
 4. Stage the file: **git add conflicted-file.ext**
 5. Commit: **git commit -m "Resolved merge conflict"**
-

Q2. JQL – Component, Labels, Sprint Filters

a. Unresolved issues in the Frontend component

JQL:

component = Frontend AND resolution = Unresolved

b. Issues labeled urgent or production-fix

JQL:

labels in (urgent, production-fix)

c. Issues in current sprint and assigned to your team

JQL:

sprint in openSprints() AND assignee in (membersOf("team-name"))

Explanation:

- Use **components** for functional groups (e.g., frontend).
 - Use **labels** for issue characteristics or context (e.g., urgent).
-

Set 5:

Q1. GitHub Scenario – Team Collaboration

(Already covered in Set 2, same steps)

Q2. JQL – Release Readiness Queries

a. Issues for **fixVersion = "v2.0"**

JQL:

fixVersion = "v2.0"

b. Unresolved bugs in v2.0

JQL:

fixVersion = "v2.0" AND issuetype = Bug AND resolution = Unresolved

c. Tasks resolved in last 7 days in v2.0

JQL:

fixVersion = "v2.0" AND resolved >= -7d

Explanation:

- **fixVersion** is used for organizing and tracking what's ready for deployment or testing.
-

Set 6:

Q1. Git Scenario – Tag and Release

Steps:

1. Tag version:
git tag v1.0
 2. Push the tag:
git push origin v1.0
 3. Go to GitHub > Releases > Create new release.
 4. Select **v1.0** tag, add release notes, and publish.
-

Q2. JQL – Overdue & SLA Queries

a. Overdue by more than 2 days

JQL:

due < -2d

b. Must be resolved within 48 hours (with SLA label)

JQL:

labels = SLA-48hr AND resolution = Unresolved

c. Issues due this week

JQL:

due >= startOfWeek() AND due <= endOfWeek()

Explanation:

JQL helps service teams enforce deadlines and track SLA-bound issues dynamically.

Set 7:

Q1. Jira Scenario – Scrum Project Setup

Steps:

1. Click **Projects > Create Project**.
 2. Select **Scrum template** and continue.
 3. After creation, go to **Backlog** and click **Create Sprint**.
 4. Add issues to backlog using **Create Issue**.
 5. Click **Start Sprint**, set sprint dates.
-

Q2. JQL – Team Assignment and Status Filters

a. Unresolved tasks assigned to Team-A

JQL:

assignee in (membersOf("Team-A")) AND resolution = Unresolved

b. Blocked issues for more than 2 days

JQL:

status = Blocked AND status CHANGED TO Blocked BEFORE -2d

c. Bugs assigned to you and in review

JQL:

assignee = currentUser() AND issuetype = Bug AND status = "In Review"

Explanation:

Use team-level filters to monitor group performance on dashboards efficiently.

Set 8:

Q1. GitHub Scenario – Fork and Contribute

Steps:

1. Go to an open-source repo > Click **Fork**.
 2. Clone it:
git clone <https://github.com/yourusername/forked-repo.git>
 3. Create new branch:
git checkout -b feature-branch
 4. Make changes, stage, and commit: **git add .**
git commit -m "Added feature"
 5. Push:
git push origin feature-branch
 6. Go to original repo and submit **Pull Request**.
-

Q2. JQL – Time-Sensitive Filters & Workload Tracking

a. **Created >= startOfWeek()**

JQL:

created >= startOfWeek()

b. **Resolved in previous quarter**

JQL:

resolved >= startOfQuarter(-1) AND resolved <= endOfQuarter(-1)

c. **Unresolved issues due in next 7 days and assigned to your team**

JQL:

**due <= 7d AND assignee in (membersOf('team-name')) AND
resolution = Unresolved**

Explanation:

- **startOfWeek()** = current week
- **startOfQuarter(-1)** = previous quarter
- **<= 7d** = due within 7 days from now

CIA 2 Batch 2

Set 1:

Q1. Git Scenario – Recover Deleted Local Changes

Steps:

1. Recover the file if it's staged:

bash

CopyEdit

git restore --staged <file>

2. Recover if it's only modified:

bash

CopyEdit

git checkout -- <file>

3. Recover if it was deleted after last commit:

bash

CopyEdit

git checkout HEAD -- <file>

Q2. JQL – Tracking Creation & Reopened Issues

a. Find issues created in the last 14 days assigned to yourself:

JQL:

bash

CopyEdit

created >= -14d AND assignee = currentUser()

b. List all issues with status Reopened after being marked Done:

JQL:

bash

CopyEdit

status = Reopened AND status CHANGED FROM Done TO Reopened

c. Find bugs created by QA team members that are unresolved:

JQL:

bash

CopyEdit

issuetype = Bug AND reporter in (QA_Team) AND resolution = Unresolved

Explanation:

- The status CHANGED operator helps in finding when an issue's status transitioned, which is key in identifying reopened issues.
-

Set 2:

Q1. GitHub Scenario – Manage Repo Access

Steps:

1. **Add a collaborator with write access:** (This is done through the GitHub web interface under "Settings" > "Manage Access.")
 2. **Prevent force pushes to main:** (This is done through the GitHub web interface under "Settings" > "Branches" > "Branch Protection Rules.")
 3. **Set up required PR reviews:** (This is done through the GitHub web interface under "Settings" > "Branches" > "Branch Protection Rules.")
-

Q2. JQL – Priority, Due Dates, and Overdue Items

a. Find all critical-priority issues assigned to your team:

JQL:

bash

CopyEdit

priority = Critical AND assignee in (team_members)

b. Show tasks due within 3 days that are unresolved:

JQL:

bash

CopyEdit

due <= 3d AND resolution = Unresolved

c. List overdue issues created in the past 20 days:

JQL:

bash

CopyEdit

created >= -20d AND due < now() AND resolution = Unresolved

Explanation:

- Using dynamic dates like <= 3d ensures that the filter adjusts to the current time, allowing for better tracking of upcoming deadlines.

Set 3:

Q1. Jira Scenario – Add Custom Workflow Status

Steps:

1. Add the status to workflow:

- Go to "Jira Settings" > "Issues" > "Workflows."
- Edit the desired workflow and click "Add Status" to add "Code Review."

2. Update transitions:

- In the workflow editor, add transitions between "In Progress" to "Code Review" and from "Code Review" to "Done."

3. Reflect changes on the board:

- Go to "Board Settings" > "Columns."
- Add the "Code Review" status to an appropriate column in your board settings.

Q2. JQL – Workflow Transition Metrics

a. Find tasks that moved to "Code Review" in the past 5 days:

JQL:

bash

CopyEdit

status CHANGED TO "Code Review" AFTER -5d

b. Show all issues where status changed from "In Progress" to "Blocked":

JQL:

bash

CopyEdit

status CHANGED FROM "In Progress" TO "Blocked"

c. List stories that have never transitioned to "Testing":

JQL:

bash

CopyEdit

issuetype = Story AND status NOT IN ("Testing")

Explanation:

- Use NOT status WAS when filtering missed transitions. This operator identifies issues that missed a particular status, such as "Testing."

Set 4:

Q1. Git Scenario – Clean Up Commits Before Push

Steps:

1. Squash commits:

bash

CopyEdit

git rebase -i HEAD~<n>

2. Rewrite the commit message:

bash

CopyEdit

git commit --amend

3. Push with force (safely):

bash

CopyEdit

git push origin <branch> --force-with-lease

Q2. JQL – Epic, Sprint & FixVersion Filters

a. Find all issues linked to epic "Checkout Redesign":

JQL:

bash

CopyEdit

"Epic Link" = "Checkout Redesign"

b. Show tasks from the current sprint that are unresolved:

JQL:

bash

CopyEdit

sprint in openSprints() AND resolution = Unresolved

c. List stories targeted for release version v1.1:

JQL:

bash

CopyEdit

issuetype = Story AND fixVersion = "v1.1"

Explanation:

- fixVersion is critical for tracking and managing releases, helping teams monitor which issues are being targeted for specific versions.
-

Set 5:

Q1. GitHub Scenario – Using GitHub Issues for Planning

Steps:

1. **Enable issues on the repo:** (This is done through the GitHub web interface under "Settings" > "Features.")
2. **Create labels like bug, enhancement, urgent:** (This is done through the GitHub web interface under "Settings" > "Labels.")
3. **Assign and close issues:**

bash

CopyEdit

git issue assign <issue_number> <assignee>

git issue close <issue_number>

Q2. JQL – Label, Component, and Reporter Filters

a. Find issues labeled security and unresolved:

JQL:

bash

CopyEdit

labels = security AND resolution = Unresolved

b. Show issues in the component API and created in the last 7 days:

JQL:

bash

CopyEdit

component = API AND created >= -7d

c. List all issues reported by devops@example.com:

JQL:

bash

CopyEdit

reporter = devops@example.com

Explanation:

- Using components over labels allows better filtering based on the issue's core functionality, making them easier to track

Set 6:

Q1. Git Scenario – Clone and Contribute to a Repo

Steps:

1. Clone the repo:

bash

CopyEdit

git clone https://github.com/<username>/<repo>.git

2. Create a branch feature-contact-form:

bash

CopyEdit

```
git checkout -b feature-contact-form
```

3. Commit and push changes:

```
bash
```

```
CopyEdit
```

```
git add <file>
```

```
git commit -m "Added contact form feature"
```

```
git push origin feature-contact-form
```

4. Submit a pull request:

- Go to the GitHub repo in your browser, navigate to "Pull Requests," and click "New Pull Request."
- Choose the base and compare branches, then create the pull request.

Q2. JQL – Bug Analysis Over Time

a. Show bugs unresolved for more than 10 days:

JQL:

```
bash
```

```
CopyEdit
```

```
issuetype = Bug AND resolution = Unresolved AND created <= - 10d
```

b. Find bugs resolved within 3 days of creation:

JQL:

```
bash
```

```
CopyEdit
```

```
issuetype = Bug AND resolutionDate <= created + 3d
```

c. Show bugs that have been reopened at least once:

JQL:

bash

CopyEdit

issuetype = Bug AND status CHANGED TO Reopened

Explanation:

- Tracking created, resolved, and status changes helps in bug analytics by identifying trends such as how quickly bugs are resolved and whether they get reopened.

Set 7:

Q1. Jira Scenario – Dashboard Customization

Steps:

1. Create a filter for unresolved bugs:

- Go to "Filters" > "Create Filter" and enter the following JQL:

bash

CopyEdit

issuetype = Bug AND resolution = Unresolved

- Save the filter.

2. Add a pie chart showing bugs by priority:

- Go to your dashboard and click "Add Gadget."
- Select "Pie Chart" and configure it to display the saved filter with "Priority" as the field.

3. Display bugs by assignee in a table view:

- Go to your dashboard and click "Add Gadget."
- Select "Filter Results" and configure it to display the saved filter. Add columns like "Assignee" and "Priority."

Q2. JQL – Assignee, Due Date, Priority Filters

a. Find all unresolved tasks assigned to yourself:

JQL:

bash

CopyEdit

assignee = currentUser() AND resolution = Unresolved

b. Show high-priority issues due this week:

JQL:

bash

CopyEdit

priority = High AND due >= startOfWeek() AND due <= endOfWeek()

c. List all issues where assignee is EMPTY:

JQL:

bash

CopyEdit

assignee IS EMPTY

Explanation:

- Unassigned tasks are important to track as they represent work that still needs to be assigned to a team member, ensuring no tasks are neglected.

Set 8:

Q1. Git Scenario – Work with Remote Branches

Steps:

1. Track the remote branch:

bash

CopyEdit

```
git branch --set-upstream-to=origin/report-gen report-gen
```

2. Pull changes made by your teammate:

```
bash
```

```
CopyEdit
```

```
git pull origin report-gen
```

3. Delete the remote branch after merge:

```
bash
```

```
CopyEdit
```

```
git push origin --delete report-gen
```

Q2. JQL – User, Status, and Date Combinations

a. Find tasks assigned to john.doe that are in progress:

JQL:

```
bash
```

```
CopyEdit
```

```
assignee = john.doe AND status = "In Progress"
```

b. Show issues created by you this month:

JQL:

```
bash
```

```
CopyEdit
```

```
creator = currentUser() AND created >= startOfMonth()
```

c. Find all issues that transitioned to "Done" after April 1, 2025:

JQL:

```
bash
```

```
CopyEdit
```


status = Done AND status CHANGED TO Done AFTER "2025-04-01"

Explanation:

- startOfMonth() is useful in tracking tasks created in the current month, ensuring that you always have up-to-date tracking for monthly reports and progress.

Retest

Set 1:

Jira / JQL Questions:

- 1. Create a Jira Cloud instance and navigate to a new project board:**
 - Go to Jira Cloud and sign up/login.
 - Click on **Create Project**.
 - Select a project template (e.g., Scrum or Kanban).
 - Follow the setup wizard and click **Create**.
 - After creation, you'll be directed to the project's board.
- 2. Write a JQL to find all issues assigned to you with status = "In Progress":**

JQL:

bash

CopyEdit

assignee = currentUser() AND status = "In Progress"

- 3. Create a filter to track high-priority issues and save it:**
 - Go to **Filters > Create Filter**.
 - Enter the following JQL:

bash

CopyEdit

priority = High

- Click **Save As** and name your filter (e.g., "High Priority Issues").
- 4. Show all issues created in the last 10 days and unresolved:**

JQL:

bash

CopyEdit

created >= -10d AND resolution = Unresolved

5. Write JQL to find all bugs resolved in the last 5 days:

JQL:

bash

CopyEdit

issuetype = Bug AND resolutionDate >= -5d

Git / GitHub Questions:

1. Initialize Git in a new local folder, create a GitHub repo, and push code:

Steps:

1. In your local folder, run:

bash

CopyEdit

git init

2. Create a new repo on GitHub (via GitHub website).
3. Link the local repo to GitHub:

bash

CopyEdit

git remote add origin https://github.com/<username>/<repo>.git

4. Add your files, commit, and push:

bash

CopyEdit

git add .

```
git commit -m "Initial commit"
```

```
git push -u origin master
```

2. **Git commands to stage, commit with message, and push to remote:**

Commands:

bash

CopyEdit

```
git add <file>      # Staging the file
```

```
git commit -m "Your message" # Commit the change
```

```
git push origin master # Push changes to GitHub
```

3. **Recover a deleted file locally using Git based on whether it was staged or modified:**

- If the file was staged:

bash

CopyEdit

```
git checkout -- <file>
```

- If the file was modified but not staged:

bash

CopyEdit

```
git checkout <file>
```

- If the file was deleted after the last commit:

bash

CopyEdit

```
git checkout HEAD -- <file>
```

Set 2:

Jira / JQL Questions:

1. Create and configure a Scrum project in Jira:

- Go to **Projects > Create Project**.
- Select **Scrum Software Development** template.
- Configure the project settings and create the project.

2. Write JQL to find issues where status changed from “To Do” to “In Progress” in the last 7 days:

JQL:

bash

CopyEdit

status CHANGED FROM "To Do" TO "In Progress" DURING (-7d, now())

3. Find overdue issues assigned to yourself using JQL:

JQL:

bash

CopyEdit

assignee = currentUser() AND due < now() AND resolution = Unresolved

4. Show unresolved tasks from the current sprint using JQL:

JQL:

bash

CopyEdit

sprint in openSprints() AND resolution = Unresolved

Git / GitHub Questions:

1. Create a GitHub repo, add a README.md, and share with a teammate:

Steps:

1. Create a new repository on GitHub.
2. In your local folder, create a README.md file.
3. Commit and push:

bash

CopyEdit

```
git add README.md
```

```
git commit -m "Add README"
```

```
git push origin master
```

2. **Create a new Git branch for a feature and merge it after completion:**

Steps:

1. Create a new branch:

bash

CopyEdit

```
git checkout -b feature-branch
```

2. Work on your feature, then commit the changes:

bash

CopyEdit

```
git add .
```

```
git commit -m "Added feature"
```

3. Merge the branch into master:

bash

CopyEdit

```
git checkout master
```

```
git merge feature-branch
```

3. **Add branch protection on main and require PR reviews:**

- Go to **Settings** of the repository.
 - Under **Branches**, add a branch protection rule for main.
 - Enable **Require pull request reviews before merging**.
-

Set 3:

Jira / JQL Questions:

1. **Add a custom status "Code Review" to a Jira workflow and update transitions:**
 - Go to **Jira Settings > Issues > Workflows**.
 - Click on the workflow, then add a new status named "Code Review".
 - Update the transitions between statuses accordingly.
2. **JQL: Find tasks that moved to "Code Review" in past 5 days:**

JQL:

bash

CopyEdit

status CHANGED TO "Code Review" DURING (-5d, now())

3. **JQL: List issues where status changed from "In Progress" to "Blocked":**

JQL:

bash

CopyEdit

status CHANGED FROM "In Progress" TO "Blocked"

4. **JQL: Find stories that never transitioned to "Testing":**

JQL:

bash

CopyEdit

issuetype = Story AND NOT status CHANGED TO "Testing"

5. JQL: Find unresolved issues due in the next 3 days:

JQL:

bash

CopyEdit

resolution = Unresolved AND due <= 3d

Git / GitHub Questions:

1. Create and push a new branch search-feature, open a pull request:

Steps:

1. Create a new branch:

bash

CopyEdit

git checkout -b search-feature

2. Make your changes and commit them:

bash

CopyEdit

git add .

git commit -m "Add search feature"

3. Push the branch:

bash

CopyEdit

git push origin search-feature

4. Open a pull request on GitHub.

2. Merge changes into the main branch after PR approval:

Steps:

1. After PR approval, merge it on GitHub via the "Merge" button.
2. Pull the changes into your local repository:

bash

CopyEdit

git checkout master

git pull origin master

Set 4:

Jira / JQL Questions:

1. **Build a Jira dashboard showing open issues, sprints, and completed tasks:**
 - Go to **Dashboards > Create Dashboard**.
 - Add gadgets like **Filter Results**, **Sprint Health**, and **Pie Chart** to display open issues, sprints, and completed tasks.
 - Configure the gadgets to display the relevant filters and data.
2. **JQL: Find unresolved issues labeled “security”:**

JQL:

bash

CopyEdit

labels = security AND resolution = Unresolved

3. **JQL: List issues created in last 7 days under API component:**

JQL:

bash

CopyEdit

component = API AND created >= -7d

4. JQL: Show issues reported by devops@example.com:

JQL:

bash

CopyEdit

reporter = devops@example.com

5. JQL: Filter issues in current sprint and assigned to your team:

JQL:

bash

CopyEdit

sprint in openSprints() AND assignee in (membersOf("Your-Team"))

Git / GitHub Questions:

1. Resolve a Git merge conflict: view, edit, and complete the merge:

Steps:

1. After pulling, Git will alert you about conflicts.
2. Open the conflicted files and manually resolve the conflicts.
3. Stage the resolved files:

bash

CopyEdit

git add <resolved-file>

4. Complete the merge:

bash

CopyEdit

git commit

2. Pull remote updates without losing local changes:

Steps:

bash

CopyEdit

git stash # Stash local changes

git pull origin master # Pull remote changes

git stash pop # Apply stashed changes back

.