Setting up development workflow

Resist the initial temptation to rush to the keyboard! This is the time to pause and think as a team about your end-to-end deployment process that you would implement in this company.

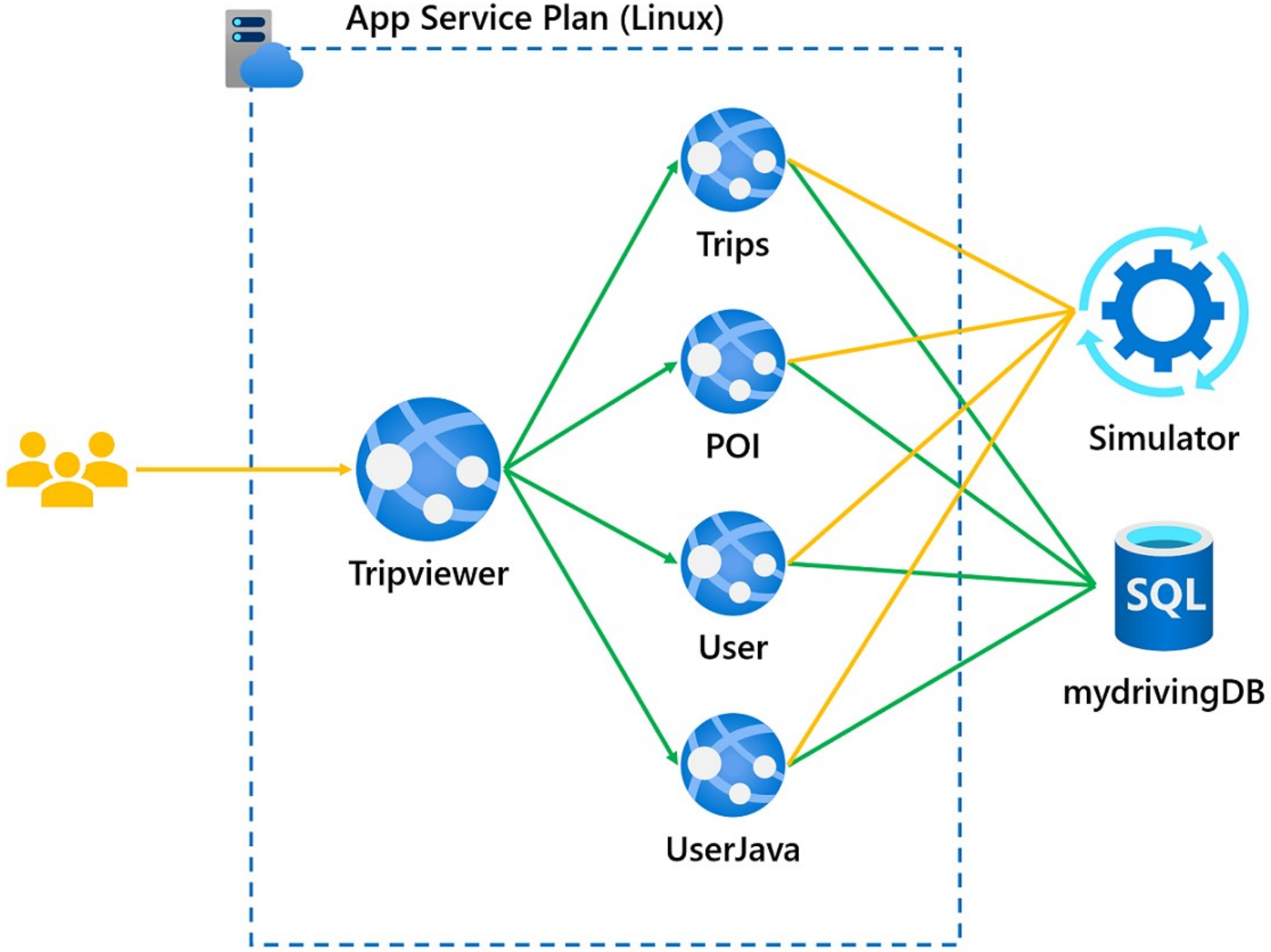
Reminders:

- Use an **InPrivate/Incognito** window in your browser to avoid any confusion with any other credentials that you may use to access Azure resources.
- The credentials you need to access the Azure subscription assigned to your team are available on the **"VIEW LAB ENVIRONMENT"** tab. Additional credentials for your teams deployed resources can be obtained following the instructions on the Overview tab under the section "Cheat sheet for the DevOps OpenHack".
- Once you have configured your Git repository you may see **security alerts** if you are an administrator in your GitHub organization. Do not be alarmed, these are intentional and you will be required to resolve security issues in a later challenge.

Prior to the event, the full solution was deployed to Azure with the following resources:

- A single Azure App Service on Linux service plan
- The team website, **Tripviewer**, that your customers are using to review their driving scores and trips which are being simulated against the APIs
- Four APIs each deployed to their own Azure Web App for Containers instance which is associated with the existing App Service plan
- A single Azure SQL server and database which host a database named **mydrivingDB**

The architecture is depicted in the following diagram:



DevOps OpenHack architecture

The code and tests for the APIs are available in the following GitHub repository:

- [Azure-Samples/openhack-devops-team (https://github.com/Azure-Samples/openhack-devops-team)](https://github.com/Azure-Samples/openhack-devops-team)

Challenge

Select the tooling that best fits your team's skills or learning plans, and configure and implement the mechanisms that will form the basis of the development workflow. You will need to set up the mechanism to link the code changes to your backlog, to prevent people from directly changing the code in your main branch without peer-reviewing (including code ownership for to establish mandatory reviewers). **You are not expected to touch the code, nor automate the build process**; this will be required in the next challenge.

Create your own code repository either by forking the provided repo on **GitHub** or import (https://help.github.com/github/importing-your-projects-to-github/importing-a-repository-with-github-importer) into **GitHub**. Your repository **must** be public.

> **Note:** Downloading and extracting an archived version of the repo locally (instead of forking) if you are following the GitHub route may be beneficial as it will narrow down the choices that your team will need to make when you are in the process of creating Pull Requests against your default branch.

If you choose to use Azure Pipelines, you will later be required to establish automatic links between a given build and its corresponding work items. Likewise, if you prefer to use GitHub Actions you will be expected to link your work items to commits as well.

You are expected to work on all the APIs of the MyDriving application.

Reminders:

- If you choose to use Azure Pipelines, be sure that when you create your Azure DevOps project, it is setup as a public project (https://docs.microsoft.com/azure/devops/organizations/public/create-public-project) to unlock the ability to use up to 10 concurrent build agents.
- The following tools have been tested with this challenge:
  - Version Control
    - **GitHub (https://github.com/)**
  - Work Management
    - **Azure Boards (https://azure.microsoft.com/services/devops/boards/)**

You are expected to implement a policy to enforce that code changes submitted are required to go through a peer review and formal approval process, since the repository contains four APIs, code ownership should be defined per API; it should be mandatory the owner(s) review the code.

Success Criteria

- Demonstrate to your coach that you have implemented a branch protection policy to prevent any code changes from being committed to the `master` branch without being reviewed. The policy must require at least two reviewers including a code owner(s) review.
- Demonstrate to your coach that you have defined code owner(s) for each of the services.
- Demonstrate to your coach that you can or have linked a work item with associated code changes.

References

A **Cheat sheet** for the DevOps OpenHack is available at the end of the **Overview** page.

> **Note**: Continuous Integration (CI) uses best practices including automated testing and ideally trunk-based development (https://trunkbaseddevelopment.com/). Automated testing will be covered later and will extend the CI setup completed here.

- What is Continuous Integration? (https://docs.microsoft.com/azure/devops/learn/what-is-continuous-integration)
- Azure DevOps Engineering Release Flow (https://docs.microsoft.com/azure/devops/learn/devops-at-microsoft/release-flow)

Work Management

- GitHub
  - Organizing work with Project Boards (https://help.github.com/github/managing-your-work-on-github/about-project-boards)
  - Managing Teams in GitHub (https://help.github.com/github/setting-up-and-managing-organizations-and-teams/about-teams)
  - Managing work with GitHub Issues (https://help.github.com/github/managing-your-work-on-github/about-issues) and GitHub Projects (https://github.com/features/project-management)
- Azure DevOps
  - Azure Boards & GitHub (https://docs.microsoft.com/azure/devops/boards/github/?view=azure-devops)
  - View and add work items using the Work Items page (https://docs.microsoft.com/azure/devops/boards/work-items/view-add-work-items)

Branch protection

- GitHub
  - GitHub - About protected Branches (https://help.github.com/articles/about-protected-branches)
  - GitHub Code Owners (https://help.github.com/github/creating-cloning-and-archiving-repositories/about-code-owners)
- Azure DevOps
  - Configure Branch Policies (https://docs.microsoft.com/azure/devops/repos/git/branch-policies)
  - Code Reviewers (https://docs.microsoft.com/azure/devops/repos/git/branch-policies?view=azure-devops#automatically-include-code-reviewers)