**Lab 1: Setting up Azure DevOps Organization & Project**

**Lab Objectives**

By the end of this lab, learners will:

1. Create an Azure DevOps Organization.
2. Create a new Project called **“DevOpsTraining”**.
3. Explore the core services (Repos, Boards, Pipelines, Artifacts, Test Plans).
4. Connect the Azure DevOps Project with an Azure Subscription for resource deployments.

**Prerequisites**

* Laptop with **Internet access**.
* Browser (latest version of Chrome/Edge recommended).
* A valid **Microsoft Account** (e.g., Outlook.com, Hotmail, or corporate Azure AD account).
* An **active Azure Subscription** with Owner/Admin access.  
  *(If not available, trainer provides Azure passes or corporate accounts based on commercials.)*

**Part A: Create Azure DevOps Organization**

1. Open browser → go to https://dev.azure.com.
   * If prompted, sign in with Microsoft account.
2. If you don’t already have an **Azure DevOps organization**, you’ll be prompted to create one:
   * Enter **Organization Name** → example: DevOpsOrg-Training
   * Select the **Region** (e.g., East US).
   * Click **Continue**.

✅ *Tip*: Region matters for compliance and latency.

1. Once created, you’ll land on the Azure DevOps **organization homepage**.

Example URL format:

https://dev.azure.com/<YourOrganizationName>

**Part B: Create New Project ("DevOpsTraining")**

1. From the organization homepage, click **New Project**.
2. Fill details:
   * **Project Name**: DevOpsTraining
   * **Description**: Training project for AZ-400 labs
   * **Visibility**: Private
   * **Version Control**: Git (default)
   * **Work Item Process**: agile
3. Click **Create**.
4. Verify you are redirected into the **DevOpsTraining** project dashboard.

**Part C: Explore Core Azure DevOps Services**

1. On the left menu bar, note the **Azure DevOps core services**:
   * **Repos** → Version control (Git) for source code.
   * **Pipelines** → CI/CD automation.
   * **Boards** → Agile project management (Work items, Sprints, Kanban).
   * **Artifacts** → Package feeds (NuGet, npm, Maven, Python).
   * **Test Plans** → Manual/automated testing management.
2. Explore each one briefly:
   * Navigate to **Repos**: See empty repo with instructions to clone or push code.
   * Navigate to **Boards**: Create a sample work item (e.g., Setup Environment).
   * Navigate to **Pipelines**: Observe “New Pipeline” wizard.
   * Navigate to **Artifacts**: View “Create feed” option.
   * Navigate to **Test Plans**: Explore test suites and cases.

**Part D: Connect Azure DevOps Project with Azure Subscription**

This step ensures Azure DevOps can deploy resources to Azure during CI/CD.

1. In the project dashboard → click on **Project Settings** (bottom left).
2. Under **Pipelines → Service connections** → click **New service connection**.
3. Select **Azure Resource Manager** → click **Next**.
4. Choose **Service principal (automatic)**.
5. Provide:
   * **Scope level**: Subscription
   * **Subscription**: Select your Azure Subscription from dropdown
   * **Service connection name**: AzureConnection-DevOpsTraining
6. Click **Save**.
7. Verify service connection is listed and accessible.

✅ *Note*: This service connection is what pipelines will use later (Terraform, deployments, AKS, etc.).

**Verification Steps**

* Confirm you can access the project at:  
  https://dev.azure.com/<OrganizationName>/DevOpsTraining
* Navigate successfully between Repos, Boards, Pipelines, Artifacts, Test Plans.
* Service connection with Azure Subscription exists and shows **“Ready”** status.

**Cleanup (Optional)**

* If this was just practice, delete the **DevOpsTraining project**:
  + Go to **Project Settings → Overview → Delete**.
* If you want to free up Azure resources, remove the **Service Connection** created.

**Expected Lab Output**

At the end of this lab, participants should have:

* An **Azure DevOps Organization**.
* A project called **DevOpsTraining** created.
* Familiarity with navigating DevOps services.
* A working **Azure Service Connection** to subscription.

**Lab 2: Configuring Agile Project Management in Azure DevOps**

**Lab Objectives**

By the end of this lab, learners will:

1. Create a **hierarchical Agile work-item structure** — Epic → Feature → User Story → Task.
2. Assign work items to specific team members.
3. Create and manage **Sprints** with backlog items.

**Prerequisites**

* **Completed Lab 1**: Have the project **“DevOpsTraining”** created.
* Access to **Azure DevOps Project** with **Basic license or above**.
* At least **two team members** added to the project (to practice assignments).
* Browser with Azure DevOps signed in:
* https://dev.azure.com/<YourOrganizationName>/DevOpsTraining

**Part A: Creating Agile Work Items**

Azure DevOps uses a hierarchy of work items that map to the Agile process template.

**1. Navigate to Boards → Work Items**

* Click the **“New Work Item”** dropdown.
* Select **Epic**.

**2. Create an Epic**

* Title: Implement E-Commerce Website
* Description: “End-to-end development and deployment of an e-commerce web application.”
* Assign to yourself or a team member.
* Click **Save & Close**.

**3. Create a Feature under the Epic**

* Open the Epic you just created.
* Scroll to **Related Work → Add link → New item → Feature**.
* Title: User Authentication and Authorization.
* Save.

**4. Create User Stories under the Feature**

* Under the Feature, click **Add link → New item → User Story**.
* Title examples:
  + As a user, I can register and log in using email/password.
  + As an admin, I can manage user roles and permissions.
* Add **Acceptance Criteria** under “Acceptance Criteria” field.
* Save each user story.

**5. Create Tasks under a User Story**

* Open the User Story → scroll to **Development → Add link → New item → Task**.
* Tasks examples:
  + Design Login Page UI
  + Integrate Azure AD B2C authentication
  + Test login and session persistence
* Save all tasks.

**Verification:**  
Your hierarchy should now look like this:

Epic: Implement E-Commerce Website

└── Feature: User Authentication and Authorization

├── User Story: Register/Login via email

│ ├── Task: Design Login Page UI

│ ├── Task: Implement backend API

│ └── Task: Validate credentials

└── User Story: Admin role management

├── Task: Create Admin Panel

└── Task: Configure RBAC

**Part B: Assign Work Items to Team Members**

1. Go to **Project Settings → Teams** → verify members are added.
   * Add additional members if needed.
2. Return to **Boards → Work Items**.
3. Open a Task → locate **Assigned To** field.
4. Select a different team member.
5. Repeat for several tasks so that everyone has something assigned.

**Part C: Create and Manage Sprints**

Sprints let you plan iterative development.

**1. Navigate to Boards → Sprints.**

* Click **New Sprint** (if not automatically generated).
* Name: Sprint 1.
* Set **Start Date** (today) and **End Date** (2 weeks later).
* Click **Save & Close**.

**2. Add Work Items to Sprint**

* Open **Backlog view (Boards → Backlogs)**.
* Drag selected User Stories or Tasks into **Sprint 1**.

**3. View Sprint Taskboard**

* Navigate to **Boards → Sprints → Taskboard**.
* Observe each work item under “To Do / Doing / Done” lanes.
* Adjust status by dragging tasks between lanes.

**Verification Checklist**

✅ Epic → Feature → User Stories → Tasks created and linked  
✅ Tasks assigned to team members  
✅ Sprint 1 created and items scheduled  
✅ Taskboard reflects progress movement

**Expected Lab Output**

Participants will leave with:

* A functioning Agile project board reflecting **real-world DevOps planning hierarchy**.
* Understanding of **linking, assigning, visualizing, and sprinting** in Azure DevOps.
* Base structure ready for upcoming CI/CD and pipeline traceability labs.

**Lab 3: Integrating Azure DevOps with External Tools**

**Lab Objectives**

By the end of this lab, learners will be able to:

1. Connect a **GitHub repository** to their Azure DevOps project.
2. Enable **Slack or Microsoft Teams notifications** for work item updates.
3. Configure **Service Hooks** for build completion events to trigger external actions.

**Prerequisites**

* Completion of **Lab 1 and Lab 2** (Azure DevOps project “DevOpsTraining” is ready).
* Access to:
  + **GitHub account** (personal or corporate).
  + **Slack workspace** or **Microsoft Teams** environment.
* Browser access to:
  + https://dev.azure.com
  + <https://github.com>
  + https://slack.com or https://teams.microsoft.com

**Part A: Connect GitHub Repository to Azure DevOps Project**

This allows your DevOps pipelines to use code hosted on GitHub instead of Azure Repos.

**Step 1: Create a GitHub Repository**

1. Log in to [GitHub](https://github.com).
2. Click **New Repository**.
   * Name: devopstraining-app
   * Description: “Sample web app for Azure DevOps integration.”
   * Visibility: Public (or Private if preferred).
   * Initialize with **README.md**.
3. Click **Create Repository**.

**Step 2: Add the GitHub Repo to Azure DevOps**

1. Go to your **Azure DevOps Project → Repos → Files**.
2. Select **Import a repository** → choose **GitHub**.
3. Sign in with your GitHub account (authorize Azure DevOps access).
4. Select your newly created repository devopstraining-app.
5. Click **Import**.

⚙️ *Behind the scenes:* Azure DevOps now mirrors the GitHub repo connection and can reference it in pipelines.

**Step 3: Verify Integration**

1. Navigate to **Pipelines → Pipelines → New Pipeline**.
2. Choose **GitHub** as your source.
3. You’ll be prompted to authorize GitHub access again (if not cached).
4. Select devopstraining-app repo.
5. Stop here — we’re not creating the pipeline yet; just ensure GitHub appears as a valid source.

✅ **Verification Checkpoint:**

* GitHub repo listed and accessible under the project.
* OAuth authorization confirmed between GitHub ↔ Azure DevOps.

**Part B: Enable Slack or Teams Notifications for Work Item Updates**

You’ll now integrate Azure DevOps notifications into a team communication channel.

**Option 1: Microsoft Teams Integration (recommended for Azure environments)**

**Step 1: Add Azure DevOps App to Teams**

1. Open **Microsoft Teams**.
2. Go to **Apps** (left sidebar).
3. Search for **Azure DevOps**.
4. Click **Add to a Team** → choose your training channel → click **Set up a connector**.

**Step 2: Configure Connector**

1. In Teams, open the channel → click the **“…” (More options)** beside the channel name.
2. Select **Connectors → Configure** under Azure DevOps.
3. Sign in with your Azure DevOps account.
4. Choose:
   * **Organization**: <YourOrganizationName>
   * **Project**: DevOpsTraining
5. Select notification types:
   * Work item created
   * Work item updated
   * Work item assigned
6. Click **Save**.

**Step 3: Test Notification**

* Go to **Azure DevOps → Boards → Work Items**.
* Create a new work item (e.g., “Sample Work Item for Teams Notification”).
* Within a few seconds, you should see a message posted in the Teams channel like:
* Work item #25 (Sample Work Item for Teams Notification) was created by <User>.

**Option 2: Slack Integration**

If you’re using Slack instead of Teams:

**Step 1: Add Azure DevOps App to Slack**

1. Go to https://slack.com/apps → search for “Azure DevOps”.
2. Click **Add to Slack**.
3. Sign in with your Slack workspace and authorize Azure DevOps.

**Step 2: Link Azure DevOps Project**

1. In any Slack channel, type:
2. /azdevops signin
3. Follow the sign-in URL and approve permissions.
4. Link the workspace to your Azure DevOps organization:
5. /azdevops link <YourOrganizationName>
6. /azdevops configure notifications
7. Choose:
   * Organization: <YourOrganizationName>
   * Project: DevOpsTraining
   * Notifications: “Work item created”, “Work item updated”.

**Step 3: Test Slack Notification**

* Create or edit a work item in Azure DevOps.
* Check Slack channel for real-time notification.

✅ **Verification Checkpoint:**

* Notifications appear in Slack or Teams channel when work items are created or updated.

**Expected Lab Output**

Participants will leave with:

* A **GitHub-linked project** for pipeline sourcing.
* **Collaborative notifications** in Slack/Teams for work items.