Azure DevOps Pipeline Configuration Guide

Objective

Use **Variable Groups** and **Task Groups** in Azure DevOps pipelines and apply branch policies and security settings to protect critical branches.

Part 1: Variable Groups in Azure DevOps Pipelines

Step 1: Create a Variable Group

1. Navigate to your Azure DevOps Project.
2. Go to **Pipelines > Library**.
3. Click on **+ Variable group**.
4. Provide a name and description for the variable group (e.g., `CommonSettings`).
5. Add key-value pairs (e.g., `environment = production`, `region = eastus`).
6. Optionally, enable the toggle **Allow access to all pipelines** or scope it to specific pipelines.
7. Click **Save**.

Step 2: Link Variable Group in YAML Pipeline

In your pipeline YAML file, link the variable group as shown below:

`yaml variables: - group: CommonSettings`

You can now access the variables like:

`yaml steps: - script: echo $(environment)`

Part 2: Set Scope for Variables in Pipeline

Step 1: Define Stage-Specific Variables

You can scope variables to specific stages directly in YAML:

```yaml stages: - stage: Build variables: buildConfiguration: 'Release' jobs: - job: BuildJob steps: - script: echo "Build Config: $(buildConfiguration)"

- stage: Deploy variables: environment: 'Production' jobs:
  - job: DeployJob steps:
    - script: echo "Deploying to $(environment)" ```

---

## Part 3: Task Groups in Azure DevOps Pipelines

### Step 1: Create a Task Group

1. Navigate to **Pipelines > Task Groups**.
2. Click **+ Create task group**.
3. Select a job/task from an existing Classic pipeline and click **Create task group**.
4. Provide a name, description, and parameterize inputs if required.
5. Click **Save**.

### Step 2: Use Task Group in Classic Pipeline

1. In your release/build pipeline, click **+ Add** in a job.
2. Search your task group by name and add it.
3. Configure parameters as needed.

   **Note**: Task Groups are supported only in Classic pipelines, not in YAML pipelines.

---

## Part 4: Apply Branch Policies and Branch Security in Azure DevOps

### Objective

Configure branch policies and security settings to protect key branches in Azure DevOps, ensuring code quality and preventing unauthorized changes.

---

### Prerequisites

- You must be a Project Administrator or have **Edit policies** and **Manage permissions** rights.
- A Git repository must be initialized in your Azure DevOps project.

---
```

Step 1: Open Azure DevOps Project

1. Navigate to https://dev.azure.com.
2. Select your organization and the target project.

Step 2: Navigate to Branches

1. Go to **Repos > Branches** from the left-hand menu.
2. Identify the branch to protect (e.g., `main` or `master`).

Part A: Apply Branch Policies

Step A1: Open Branch Policies

1. Click the **three-dot menu (⋮)** next to the target branch.
2. Select **Branch policies**.

Step A2: Configure the Following Policies

- **Minimum number of reviewers**: Require at least one reviewer before completing a PR.
- **Check for linked work items**: Require linking to work items.
- **Check for comment resolution**: Ensure all comments are addressed.
- **Limit merge types**: Choose allowed merge types (e.g., squash, rebase).
- **Build validation**:
  - Add a build pipeline that must succeed before merging.
- **Automatically include reviewers**: Automatically assign teams/users as reviewers.

Step A3: Save Policy

Click **Save changes** after configuring.

Part B: Apply Branch Security

Step B1: Open Branch Security Settings

1. In the **Branches** view, click the **three-dot menu (⋮)** for the branch.
2. Choose **Branch security**.

Step B2: Configure Permissions per Group/User

You will see permissions for various groups like: - **Project Administrators** - **Contributors** - **Readers**

Step B3: Common Permission Settings

Permission	Admins	Contributors	Readers
Contribute	Allow	Deny	Deny
Force push (rewrite history)	Allow	Deny	Deny
Create branch	Allow	Allow	Deny
Delete	Allow	Deny	Deny
Manage permissions	Allow	Deny	Deny

1. Select the group (e.g., Contributors).
2. Set `Contribute`, `Force push`, and other sensitive actions to **Deny**.
3. Set appropriate **Allow** for Project Administrators.
4. Click **Save changes**.

Step C: Validate Security and Policies

- Attempt a direct push to the protected branch from a Contributor account: should be blocked.
- Create a pull request to merge into the branch: should enforce the configured policies.

Best Practices

- Apply branch policies to all production branches (`main`, `release/*`).
- Deny force-push for all non-admin roles.
- Use build validation for automated quality checks.
- Regularly audit branch security settings.

References

- Variable Groups in Azure Pipelines
- Task Groups
- Branch Policies
- Branch Permissions