3. Build and release pipelines
└ **3.1 Advanced pipeline design**
└ **3.3.2 Reusable elements: YAML templates, task/variable groups**

1. What are YAML templates in Azure Pipelines, and why use them?
2. How do you create and include a template in a pipeline YAML file?
3. What is a task group, and how is it different from a YAML template?
4. What are variable groups, and how are they used in Azure Pipelines?
5. How do you reference variable groups in YAML pipelines?
6. How can you pass parameters to YAML templates?
7. What is the best practice for sharing templates across multiple pipelines or repos?
8. How do you manage secrets with variable groups?
9. How do you use conditions and expressions within YAML templates?
10. What are common pitfalls or limitations when using reusable pipeline elements?

1. What are YAML templates in Azure Pipelines, and why use them?
YAML templates are reusable pipeline definitions that allow you to define pipeline logic once and include it in multiple pipelines. They promote DRY principles and consistency across builds and releases.

2. How do you create and include a template in a pipeline YAML file?
Define the template in a separate YAML file (e.g., template.yml). Include it in your pipeline using the template keyword:
```
- template: template.yml
```
Parameters can be passed if defined.

3. What is a task group, and how is it different from a YAML template?
A task group is a saved set of tasks from classic (visual designer) pipelines, reusable in multiple pipelines via the UI. YAML templates are code-based and used in YAML pipelines for reusability and version control.

4. What are variable groups, and how are they used in Azure Pipelines?
Variable groups store and manage values (e.g., connection strings, secrets) shared across pipelines. They centralize variables for reuse and easier updates.

5. How do you reference variable groups in YAML pipelines?
Use the variables block and the group keyword:
```
variables:
  - group: MyVariableGroup
```
This makes the group's variables available in the pipeline.

6. How can you pass parameters to YAML templates?
Define parameters in the template using the parameters block. Pass values when including the template:
```
- template: template.yml
  parameters:
    name: value
```

7. What is the best practice for sharing templates across multiple pipelines or repos?
Store templates in a central repository, reference them via repository resources, and use versioning or branch references for control.

8. How do you manage secrets with variable groups?
Mark variables as “secret” in the variable group. Access secrets in pipelines but they are masked in logs and not exposed to scripts by default.

---

9. How do you use conditions and expressions within YAML templates?
Templates and steps can use condition and expressions (e.g., `eq(variables['Build.SourceBranch'], 'refs/heads/main')`) to control execution based on pipeline context or parameters.

---

10. What are common pitfalls or limitations when using reusable pipeline elements?
Common issues:

- Parameter type mismatches
- Circular template references
- Scope/visibility problems with variable groups
- Inability to use task groups in YAML pipelines (YAML and task groups are separate)
- Secret variables not available to all steps by default