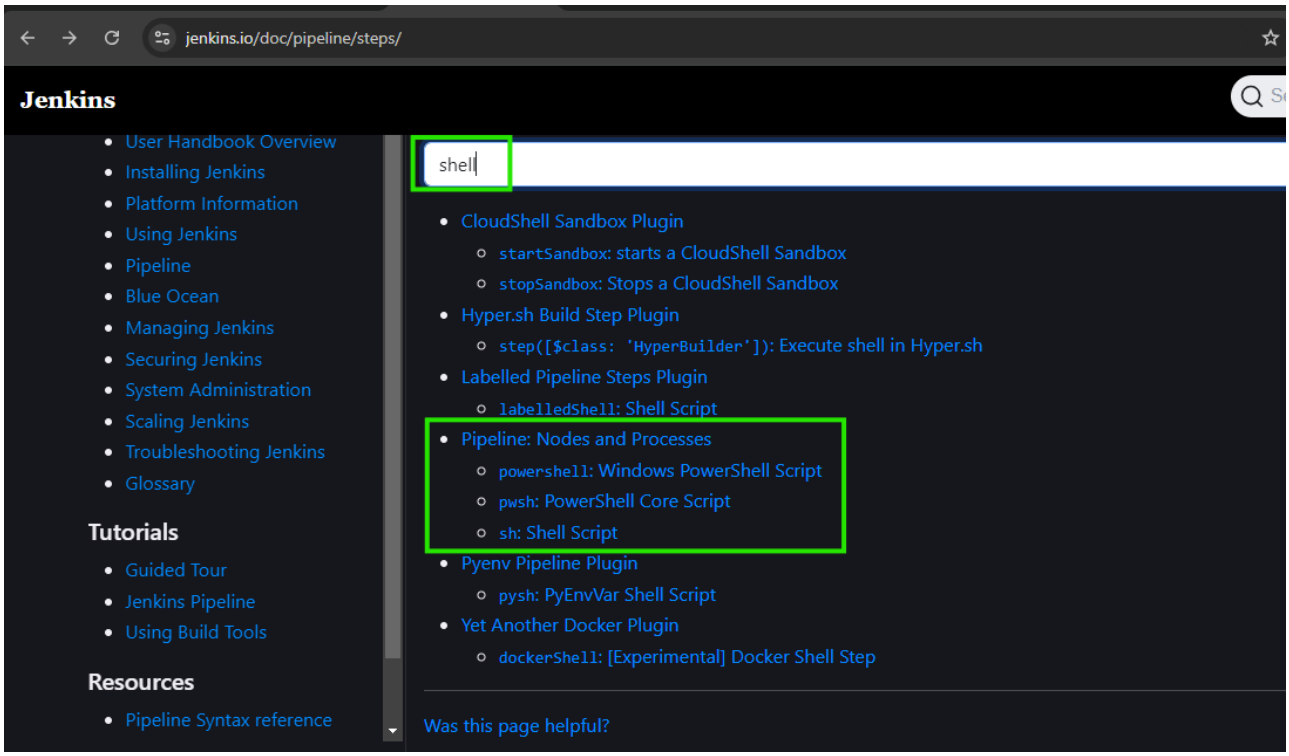
Plugins w.r.t Pipelines

- For Pipelines, Plugins provide extra steps
- Refer Here for artifactory plugin and the steps added by plugin

Ways of Working (WOW) with Pipeline in Jenkins

- Ensure you have all the manual build steps
- Express manual steps in the Pipeline Syntax Refer Here
- For each manual step try finding the step Refer Here.
- If the plugin is not installed



Apply the above (WOW) for the dotnet project

- Manual Steps
  - Installing dotnet Refer Here
  - Clone the nopCommerce project Refer Here
  - Restore nuget packages

```
dotnet restore src/Presentation/Nop.Web/Nop.Web.csproj
```

```
ubuntu@ip-172-31-40-237:~/nopCommerceaug24$ dotnet restore src/Presentation/Nop.Web/Nop.Web.csproj

Welcome to .NET 8.0!
---------------------
SDK Version: 8.0.108

----------------
Installed an ASP.NET Core HTTPS development certificate.
To trust the certificate, view the instructions: https://aka.ms/dotnet-https-linux

----------------
Write your first app: https://aka.ms/dotnet-hello-world
Find out what's new: https://aka.ms/dotnet-whats-new
Explore documentation: https://aka.ms/dotnet-docs
Report issues and find source on GitHub: https://github.com/dotnet/core
Use 'dotnet --help' to see available commands or visit: https://aka.ms/dotnet-cli
--------------------------------------------------------------------------------------
  Determining projects to restore...
```

- Build (Debug/Release) Refer Here

```
dotnet build --configuration Release src/Presentation/Nop.Web/Nop.Web.csproj
```

- Now we need to package, we will be performing publish and provide output folder Refer Here

```
dotnet publish --configuration Release --output published/
src/Presentation/Nop.Web/Nop.Web.csproj
```

- Lets build a CI Pipeline for a day build
  - whenever any change is done on develop branch
  - restore
  - build
  - test (in this case we are skipping for nopcommerce)
  - create package
- Refer Here for the changes to implement CI Pipeline

Day build and Nightly build

- Day builds are builds generated during Active development time
- Here we need to give feedback to the developer about his latest commit
  - we build the code to create package
  - we run minimal tests
    - smoke test
    - sanity test
    - integration test
- Nightly build is the build generated during non working times
- Here we build and run end to end tests * selenium (UI Based) * Jmeter (Performance testing) * Penetration testing * stress testing

Tests of importance

Smoke tests, sanity tests, integration testing, and regression testing are essential components in the software development and DevOps lifecycle. Each serves a unique purpose in ensuring software quality and stability.

Smoke Testing

Purpose

Smoke testing, also known as build verification testing, is a preliminary test to check whether the most crucial functionalities of an application are working correctly after a new build. It aims to identify severe failures early in the development process, allowing teams to determine if the software is stable enough for further testing.

Characteristics

- **Broad Coverage**: Focuses on critical functions without delving into finer details.
- **Quick Execution**: Typically runs quickly, often after every build or significant code change.
- **Automation**: Can be automated for efficiency, though manual tests are also common.

Benefits

- Detects major issues early, reducing the risk of larger problems later.
- Provides fast feedback, which is vital for continuous integration and deployment (CI/CD) pipelines[1][3][4].

Sanity Testing

Purpose

Sanity testing is a subset of regression testing that verifies specific functionalities after changes have been made to the codebase. It ensures that the new code changes do not adversely affect existing features.

Characteristics

- **Narrow Focus**: Concentrates on specific functionalities rather than the entire application.
- **Quick Checks**: Performed after minor changes or bug fixes to confirm that the issues have been resolved.

Benefits

- Quickly validates that specific changes work as intended, allowing for faster iterations in development.

Integration Testing

Purpose

Integration testing evaluates the interaction between different modules or components of an application. It ensures that integrated parts work together as expected.

Characteristics

- **Component Interaction**: Tests how different modules interact and communicate.
- **Varied Approaches**: Can be performed using various strategies, including bottom-up, top-down, and sandwich approaches.

Benefits

- Identifies interface defects between integrated components, which can be critical for overall system functionality.
- Helps ensure that combined parts of the application function correctly as a cohesive unit.

Regression Testing

Purpose

Regression testing is performed to confirm that recent changes or enhancements in the code have not introduced new bugs into existing functionality. This type of testing is crucial after updates or bug fixes.

Characteristics

- **Comprehensive Coverage**: Involves re-running previously completed tests to ensure unchanged parts of the application still work.
- **Automation Friendly**: Often automated due to the repetitive nature of the tests.

Benefits

- Provides confidence that new code changes do not disrupt existing functionality.
- Essential for maintaining software quality over time, especially in agile development environments where changes are frequent.

In summary, these testing methodologies are integral to the DevOps process, ensuring that software is robust, functional, and ready for deployment. Each type of testing serves a specific role in the quality assurance process, contributing to the overall reliability of the software being developed.