Objective2: Lay down a plan for CI/CD. As there will be tests pertaining to each microservice, how do you plan to run them in tandem in the deployment pipeline.

Microservices must continue to work when the rest of the world shuts down, so the following consequence is right- it must be built and tested separately first. The pipeline for one micro services would be similar to the one depicted below.

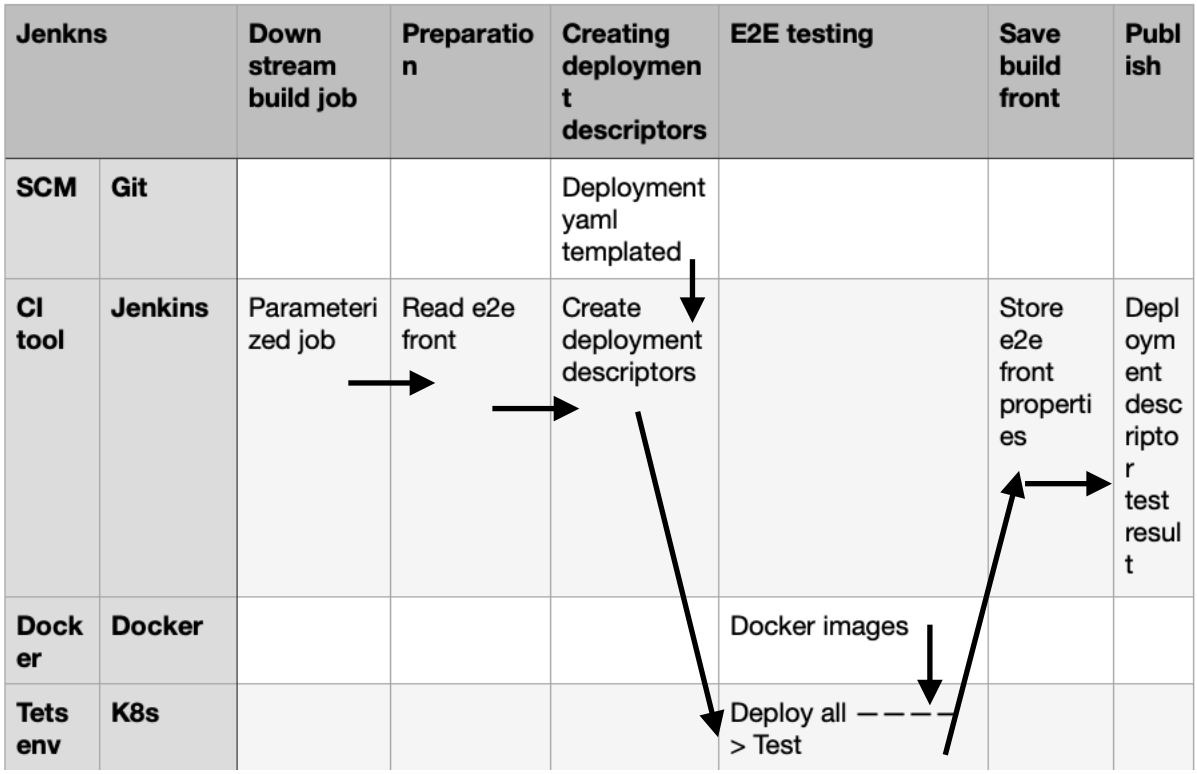
Jenkins		Preparation	Build and Unit testing	Publish	Deploy	Integration testing	Publish	Trigger end to end pipeline
SCM	**Git**	Checkout source code, create version			Helm			
Build tool	**Maven**		Build, unit and component testing					
CI tool	**Jenkins**			Publish docker images into temporal registry	Create deployment descriptor			Start end to end testing
Docker registry	**Docker**						Publish to Docker registry	
Dev Env	**K8s**				Deploy	Regression and integration testing		
Test Env	**K8s**				Deploy			

- Merging code changes into the master/build branch in SCM will trigger the build procedure.
- Unit and component tests are performed during the build procedure. Static code analysis is also performed to pass the source code quality gate.
- Publish the build artifact into the temporal repository.
- Create temporal deployment descriptors from templates. Velocity, Groovy, or any other template can be used in case of Jenkins.

- Deploy to the development environment, just to allow manual checks for developers regardless of integration testing result.
- Deploy to the test environment and start integration testing. It's nice to have externalized test scenarios and procedures.
- Publish into a repository in case integration testing is passed. Trigger integration with other microservices.

NOTE: The end to end testing will be having the latest stable version of other micro services.

End to end testing pipeline



Steps:

- The downstream build sends the names and version of microservices which have been changed.

- The build server loads the latest version numbers of all existing microservices. The simple property file is enough.
- Merge versions with the new one. The set of the versions and deployment templates are used to create deployment descriptors.
- Deploy microservices using a set of deployment descriptors with the changed version and run integration tests.
- If the testing procedure passes, the build server will persist the set of microservice versions as to use next time.
- Compose configuration and publish them. To simplify the deployment procedure, it makes sense to create one deployment descriptor, which will include all necessary microservices, configuration variables, etc.
-