Name : M. Sahithi

H.NO : 22C11A05B5

Class : III B.Tech

Section : CSE-B

Subject : DevOps

1) Compare and constrast the Agile development model with the traditional waterfall model.

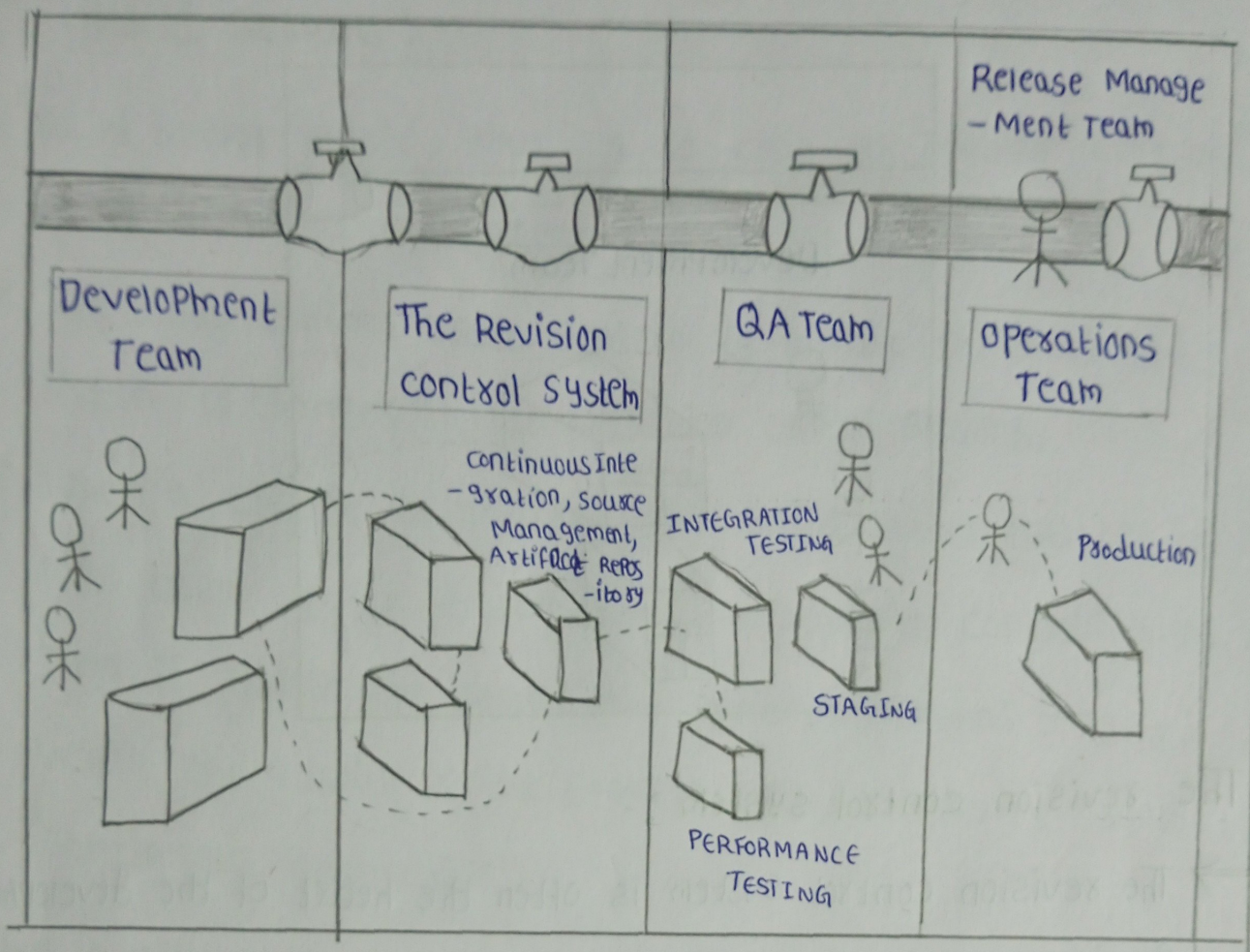
Agile and waterfall model are two different methods for the software development process. Despite their differences in approach, both methodologies can be used at times, depending upon the project and the requirements.

Agile Model	Waterfall Model
1) Agile Methodologies propose incremental and iterative approaches to software design	1) Software development flows sequentially from start point to end point.
2) The Agile Model in software engineering is broken into individual models that designe -rs work on	2) The design process is not broken into individual models
3) The customer has early and frequent opportunities to look at the Product and make decisi -on and changes	3) The customer can see the product at the end of the project.
4) Test Plan is reviewed after each sprint.	4) The test Plan is hardly discussed during the test Phase.

Agile Model	Waterfall Model
5) The Agile Model is considere -d unstructured compared to the waterfall Model	5) waterfall model are more secure because they are Plan oriented.
6) Small Projects can be implemented very quickly. For large Projects, it isn't easy to estimate the develop -ment time.	6) All sorts of the Project can be estimated and completed.

(2) Explain the concepts of Continuous Delivery and its role in the DevOps Process.

- When we work with Devops, we work with large and complex processes in a large and complex context.
- A Continuous Delivery Pipeline in a large organization is introduced in the following image:



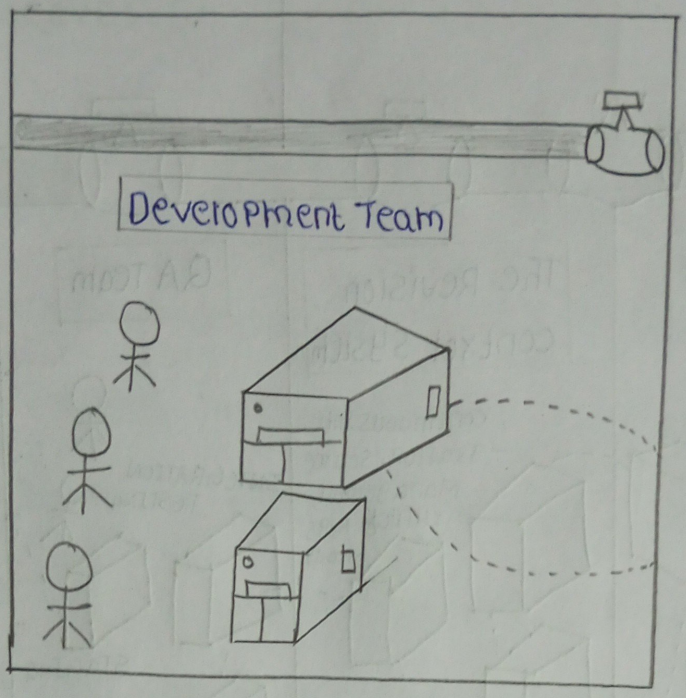
- While the basic outline of this image holds true surprisingly often, regardless of the organization.
- There are, of course, differences, depending on the size of the organization and the complexity of the Products that are being developed.

The developers

→ The developers work on their stations.

⇒ They develop code and need many tools to be efficient.

⇒ If we have a strong developer background, we appreciate the convenience of a prepackaged developer environment.

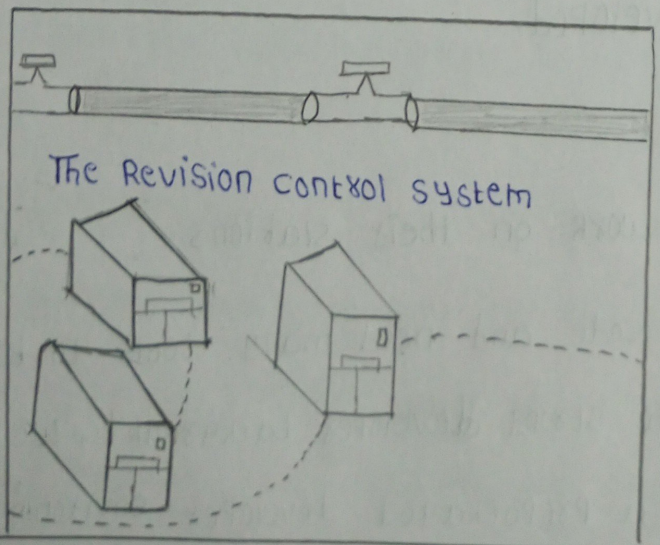


The revision control system :-

⇒ The revision control system is often the heart of the development environment.

⇒ The code that forms the organization's software products is stored here.

⇒ It is also common to store the configurations that form the infrastructure here.



The Build Server :-

⇒ "Build server builds your source code at regular intervals" or on different triggers.

⇒ The most common usage pattern is to have the build server listen to changes in the revision control system.

⇒ When a change is noticed, the build server updates its local copy of the source from the revision control system. Then, it builds the source and performs optional tests to verify the quality of the changes. This process is called "continuous integration."

The artifact Repository :-

⇒ When the build server has verified the quality of the code and compiled it into deliverables, it is useful to store the compiled binary artifacts in a repository.

⇒ This is normally not the same as the revision control system.

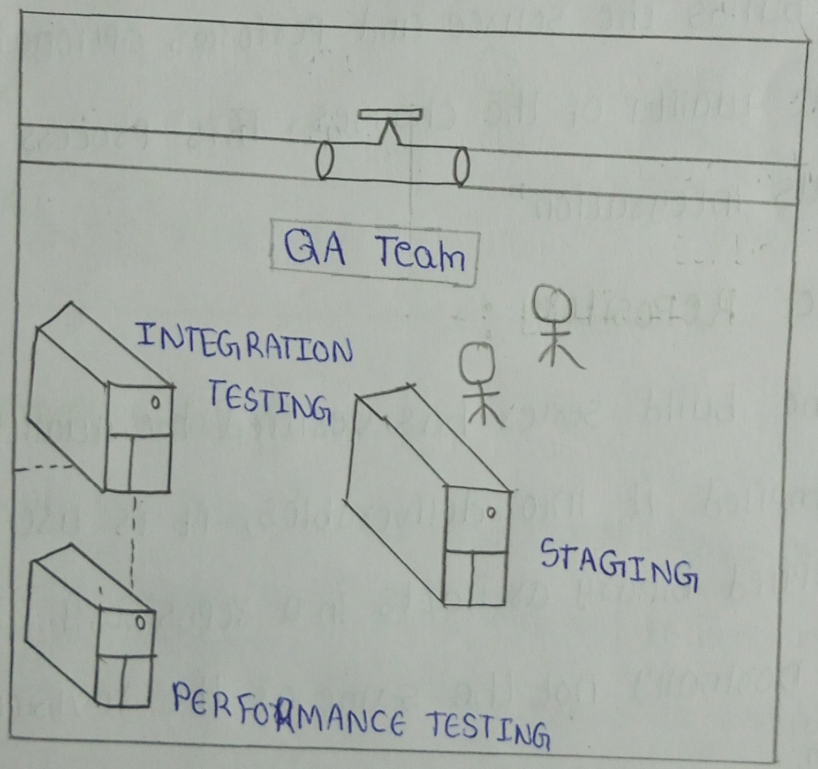
Package Managers :-

⇒ Linux servers usually employ systems for deployment that are similar in principle but have some differences in practice. "Red-Hat-like" systems use a package format called RPM.

⇒ The deliverables can then be installed on servers with a command that fetches them from a binary repository. These commands are called "Package Managers".

Test Environments :-

⇒ After the build server has stored the artifacts in the binary repository, they can be installed from there into test environments.



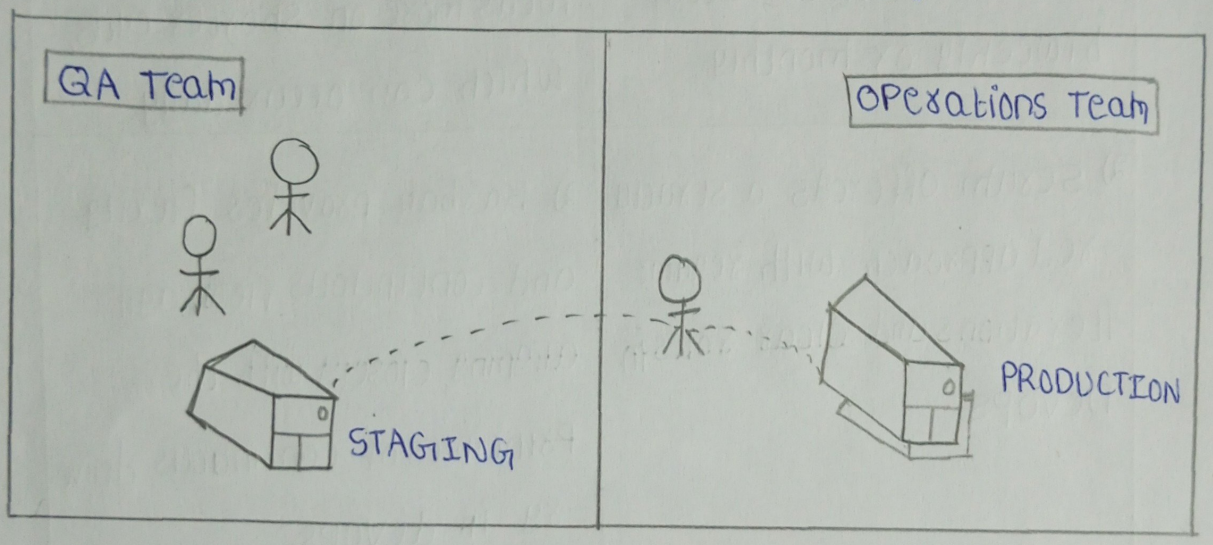
⇒ Test environments should normally attempt to be as production-like as is feasible.

⇒ Therefore, it is desirable that the they be installed and configured with the same methods as production servers.

Staging / Production :-

⇒ Staging environments are the last line of test environments. They are interchangeable with production environments.

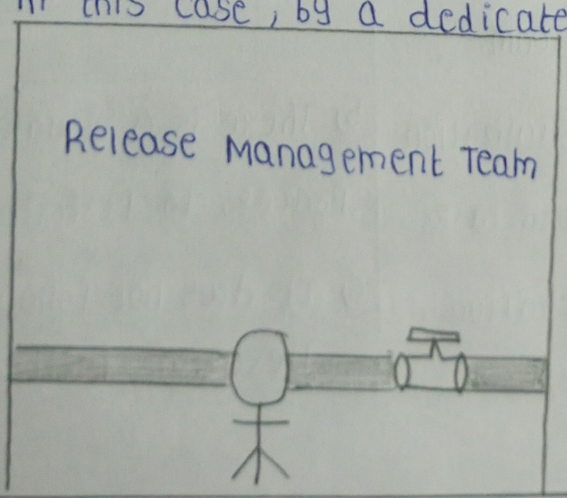
⇒ The following detail from the larger continuous delivary image shows the final systems and roles involved:



Release Management :-

⇒ one reason for this is that it is usually hard to reach the level of test automation needed in order to have comple confedence in automated deploys.

⇒ A fault is used in the following figure to symbolize human interaction – in this case, by a dedicated release manager.



3) Compare and contrast the Scrum and Kanban methodologies in the context of Devops.

Scrum	Kanban
1) Scrum focuses on sprint cycles, which can occur biweekly or monthly.	1) Kanban can be said to focus more on shorter cycles, which can occur daily.
2) Scrum offereds a structu-red approach with regular iterations and clear roles in Devops.	2) Kanban provides flexity and continuous flow, ~~agil~~ aligning closely with the principles of continuous delive-ry in devops.
3) scrum's fixed-length Sprints.	3) Kanban's continuou flow.
4) Scrum roles: Product owner, Scrum master, scrum Team.	4) No required skills.
5) There is no visualization process to perform tasks.	5) There is a visualization process to perform tasks.
6) It follows iterative method.	6) It does not follow the iterative method.