Task(4)

DevOps life cycle:

* Discover
* Plan
* Build
* Test
* Monitor
* Operate
* Continuous feedback

## 1- Discover:

In the Discover phase, a DevOps team researches and defines the scope of a project. In particular, it involves activities such as user research, establishing goals, and defining success.

Tools like:

Mural and Miro empower the entire software team to share ideas and conduct research. [Jira Product Discovery](https://www.atlassian.com/software/jira/product-discovery) organizes this information into actionable inputs and prioritizes actions for development teams. As you’re prioritizing, you’ll also need to keep your backlog of user feedback in mind.

Product discovery is the very first activity of a product design, which then becomes the baseline for decision making

* 2- Plan:

Taking a page out of the agile handbook, we recommend tools that allow development and operations teams to break work down into smaller, manageable chunks for quicker deployments. This allows you to learn from users sooner and helps with optimizing a product based on the feedback. Look for tools that provide sprint planning, issue tracking, and allow collaboration, such as Jira.

Another great practice is continuously gathering user feedback, organizing it into actionable inputs, and prioritizing those actions for your development teams. Look for tools that encourage “asynchronous brainstorming” (if you will). It’s important that everyone can share and comment on anything: ideas, strategies, goals, requirements, roadmaps and documentation.

3- Build:

developers use open source tools like Kubernetes and Docker to provision individual development environments. Coding against virtual, disposable replicas of production helps you get more work done.

### **Infrastructure as code:**

1-chef

2-doker

3-puppet

4-terra form

[Infrastructure as code](https://www.atlassian.com/continuous-delivery/principles/infrastructure-as-code?utm_campaign=service-desk_devops16-blog) means re-provisioning is faster than repairing – and more consistent and reproducible. It also means you can easily spin up variations of your development environment with similar configuration as production. Provisioning code can be applied and reapplied to put a server into a known baseline. It can be stored in [version control](https://www.atlassian.com/git/tutorials/what-is-version-control). It can be tested, incorporated into [CI (continuous integration)](https://www.atlassian.com/continuous-delivery/continuous-integration), and peer-reviewed.

### **Source control and collaborative coding:**

1-githup

2-bit bucket

Source control tools help store the code in different chains so you can see every change and collaborate more easily by sharing those changes. Rather than waiting on change approval boards before deploying to production, you can improve code quality and throughput with peer reviews done via pull requests.

4-continuous delivery:

1-jenkin

2-AWS

3-Bit bucket

4-circle ci

Continuous integration is the practice of checking in code to a shared repository several times a day, and testing it each time. That way, you automatically detect problems early, fix them when they’re easiest to fix, and roll out new features to your users as early as possible.

Code review by pull-requests requires branching and is all the rage. The DevOps North Star is a workflow that results in fewer and faster branches and maintains testing rigor without sacrificing development speed.

4- Test :

Testing tools span many needs and capabilities, including exploratory testing, test management, and orchestration. However, for the DevOps toolchain, automation is an essential function. [Automated testing](https://www.atlassian.com/devops/devops-tools/test-automation) pays off over time by speeding up your development and testing cycles in the long run. And in a DevOps environment, it’s important for another reason: awareness.

Test automation can increase software quality and reduce risk by doing it early and often. Development teams can execute automated tests repeatedly, covering several areas such as UI testing, security scanning, or load testing. They also yield reports and trend graphs that help identify risky areas.

6-operate:

### **Incident, change and problem tracking**

1-Jira service management

2-Jira software

3-status page

The keys to unlocking collaboration between DevOps teams is making sure they’re viewing the same work. What happens when incidents are reported? Are they linked and traceable to software problems? When changes are made, are they linked to releases?

Nothing blocks Dev’s collaboration with Ops more than having incidents and software development projects tracked in different systems. Look for tools that keep [incidents](https://www.atlassian.com/incident-management), [changes](https://www.atlassian.com/itsm/change-management), [problems](https://www.atlassian.com/itsm/problem-management), and software projects on one [platform](https://www.atlassian.com/software/jira/service-management) so you can identify and fix problems faster.

7-observe:

1-app dynamic

2-spacks

3-splug

4-new relic

5-ping Dom

There are two types of monitoring that should be automated: server monitoring and application performance monitoring.

Manually “topping” a box or hitting your API with a test is fine for spot-checking. But to understand trends and the overall health of your application (and environments), you need software that is listening and recording data . Ongoing observability is a key capability for successful DevOps teams.

Look for tools that integrate with your group chat client so alerts go straight to your team’s room, or a dedicated room for incidents

8- continuous feedback:

1-pendo

2-slack

3-get feedback

Customers are already telling you whether you’ve built the right thing – you just have to listen. Continuous feedback includes both the culture and processes to collect feedback regularly, and tools to drive insights from the feedback. Continuous feedback practices include collecting and reviewing NPS data, churn surveys, bug reports, support tickets, and even tweets

2-compare between DevOps and Agile:

|  |  |  |
| --- | --- | --- |
| Point of comparison | Agile | DevOps |
| definition | Agile is a philosophy about how to develop and deliver software | DevOps describes how to continuously deploy code through the use of modern tools and automated processes. |
| inception | 2001 | 2007 |
| Highest priority | Continuous delivery of software | Continuous deployment of software |
| Founding artifact | Agile Manifesto | 10+ Deploys Per Day conference session and The Phoenix Project |
| Implementation frameworks | Scrum, Kanban, ScrumBan, Lean, XP | CAMS, [CALMS](https://www.techtarget.com/searchitoperations/tip/Use-the-CALMS-framework-for-a-smooth-DevOps-transition), DORA |
| Alternatives | Waterfall | Silo based development and deployment |
| Team size | Used by small teams of 10 or less | Implemented as a company wide strategy |
| Scope | Focuses on the development of a single application | A corporate wide approach to software deployment |

3-what are we know about Data Ops and Ml Ops?

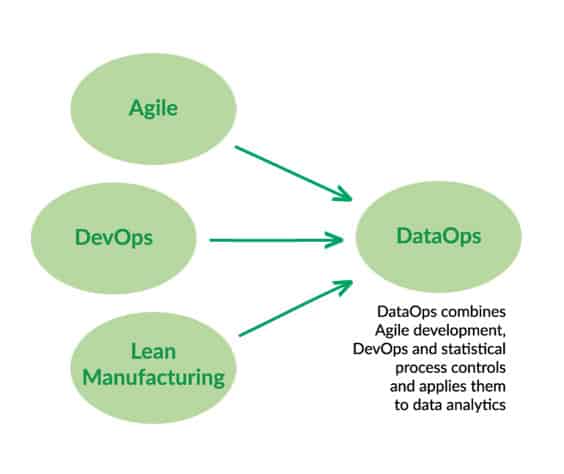
Data Ops: the term “DataOps” does communicate that data analytics can achieve what software development attained with [DevOps](https://bit.ly/2l0Tk9k). That is to say, DataOps can yield an order of magnitude improvement in quality and cycle time when data teams utilize new tools and methodologies. DevOps optimizes the software development pipeline. It is what allows companies like Amazon, Netflix and Google to execute millions of code releases per year

DataOps also accelerates software (new analytics) development but has to simultaneously manage a dynamic manufacturing operation (i.e., data operations). DataOps includes DevOps and other methodologies which apply to the unique challenges of managing an enterprise-critical data operations pipeline

**It is a collection of technical practices, workflows, cultural norms, and architectural patterns that enable**

* Rapid innovation and experimentation delivering new insights to customers with increasing velocity
* Extremely high data quality and very low error rates
* Collaboration across complex arrays of people, technology, and environments
* Clear measurement, monitoring, and transparency of results

The link between Data Ops ,Dev Ops andAgile:



DataOps applies Agile development, DevOps and lean manufacturing to data analytics development and operations. Agile is an application of the Theory of Constraints to software development, i.e., smaller lot sizes decrease work-in-progress and increase overall manufacturing system throughput. DevOps is a natural result of applying lean principles (e.g., eliminate waste, continuous improvement, broad focus) to application development and delivery. Lean manufacturing also contributes a relentless focus on quality, using tools such as statistical process control, to data analytics.

MlOps:

MLOps is a useful approach for the creation and quality of machine learning and AI solutions. By adopting an MLOps approach, data scientists and machine learning engineers can collaborate and increase the pace of model development and production, by implementing continuous integration and deployment (CI/CD) practices with proper monitoring, validation, and governance of ML models.

Productionizing machine learning is difficult. The machine learning lifecycle consists of many complex components such as data ingest, data prep, model training, model tuning, model deployment, model monitoring, explainability, and much more. It also requires collaboration and hand-offs across teams, from Data Engineering to Data Science to ML Engineering.

The benefit of MlOps:

The primary benefits of MLOps are efficiency, scalability, and risk reduction. Efficiency: MLOps allows data teams to achieve faster model development, deliver higher quality ML models, and faster deployment and production. Scalability: MLOps also enables vast scalability and management where thousands of models can be overseen, controlled, managed, and monitored for continuous integration, continuous delivery, and continuous deployment.

The Data science step for ML:

1. Data extraction: You select and integrate the relevant data from various data sources for the ML task.
2. Data analysis: You perform [exploratory data analysis](https://wikipedia.org/wiki/Exploratory_data_analysis) (EDA) to understand the available data for building the ML model. This process leads to the following:
   * Understanding the data schema and characteristics that are expected by the model.
   * Identifying the data preparation and feature engineering that are needed for the model.
3. Data preparation: The data is prepared for the ML task. This preparation involves data cleaning, where you split the data into training, validation, and test sets. You also apply data transformations and feature engineering to the model that solves the target task. The output of this step are the *data splits* in the prepared format.
4. Model training: The data scientist implements different algorithms with the prepared data to train various ML models. In addition, you subject the implemented algorithms to hyperparameter tuning to get the best performing ML model. The output of this step is a trained model.
5. Model evaluation: The model is evaluated on a [holdout test set](https://wikipedia.org/wiki/Training,_validation,_and_test_sets#Holdout_dataset) to evaluate the model quality. The output of this step is a set of metrics to assess the quality of the model.
6. Model validation: The model is confirmed to be adequate for deployment—that its predictive performance is better than a certain baseline.
7. Model serving: The validated model is deployed to a target environment to serve predictions. This deployment can be one of the following:
   * Microservices with a REST API to serve online predictions.
   * An embedded model to an edge or mobile device.
   * Part of a batch prediction system.
8. Model monitoring: The model predictive performance is monitored to potentially invoke a new iteration in the ML process.