**Task 1**

**Search for Data ops and what is the differance between Dataops and Devops?**

**What is the use of dataops in speeding up the pipeline and solving any problem with ai??**

DataOps is designed to build high quality data and analytics solutions at an increasingly accelerated pace, and with higher reliability, as time goes on.

As organizations have struggled beneath a deluge of data, their data teams faced growing expectations that the business put that data to work. Data teams were inspired by the DevOps methodology to create DataOps.

DataOps was created to leverage the underlying manufacturing methodologies of lean manufacturing, statistical process control, and, of course, agile development.

DataOps seeks to quickly find the right data for the right application. It brings together business users, data scientists, data analysts, IT, and application developers to fulfill the business need for insights. DataOps then works to continuously improve and adjust data models, visualizations, reports, and dashboards to achieve business goals.

DataOps fosters cross-functional collaboration and automation to build fast, trustworthy data pipelines so your business can wring the most value from your data.

**differences between DataOps vs DevOps**

differences between DataOps vs DevOps are the ultimate goals of the different components of the development cycles. For example, though both DataOps and DevOps have a quality component, DevOps aims to create a quality end product, while DataOps must ensure high quality data enters the process and trusted, high quality outputs are used appropriately by the business. Other nuanced differences between DataOps vs DevOps are:

Quality: DataOps ensures usage of high quality data for high quality outputs; DevOps delivers a quality product.

Collaboration: DataOps works with business users, application developers, and IT operations; DevOps works with engineering and development teams.

Cycle Times: DataOps strives to build a continuous data pipeline so business users become self-sufficient; DevOps strives for shorter release cycles to meet business demands.

Operations: DataOps is constantly addressing new and changing data challenges involving many sources and needs; DevOps runs repeatable, highly similar cycles.

**Task 2**

**What is the use of integration in ai?**

Integration is all about the connecting and moving of data, so that it can be safely stored and used to help you run your business and make decisions. Good integration distributes good data through your systems: if you think of data as cars, integration is like the road network:  if it’s done wrong, there are traffic jams, wrong turns and lots of frustration.

You might wonder how this is related to artificial intelligence (AI) and machine learning (ML).

Again, it’s all down to data. AI and ML might sound futuristic and sci-fi but the hardest part of getting value from these technologies comes from supplying them with good training and testing data.

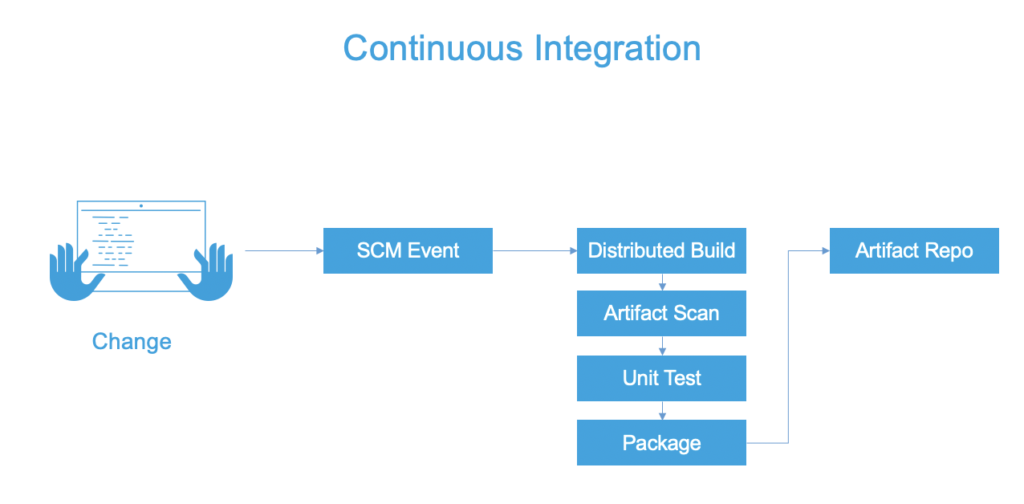
And seeing as good integration means a good flow of data, businesses with good integration systems are well placed to enhance their data processing via AI and ML.

**Task 3**

**What is the differance between Integration and Deployment and what is the best in terms of performance?**

**Continuous Integration:**

Continuous Integration is the automation of builds. Depending on your source control/version control strategy, code changes for a bug fix or new feature need to be merged/committed into a branch in the source code repository. No matter which side of the mono-repo vs. multi-repo argument you are on, a build – and eventually, release artifacts – will be created as part of the Continuous Integration processes.



**Why it’s Important**

Rarely do you work alone as a software engineer. Integrating your features/bug fixes into the application is a common task for a software engineer. For the newly-minted software engineer whose only experience is in group projects, that can take a little getting used to. The ability to merge ideas quickly is the big allure of Continuous Integration. With modern systems, the build and packaging steps can be different. In JAVA development, for example, the JAVA build produces a JAR file. Then that JAR file is packaged into a Docker Image for deployment.

Three pillars that Continuous Integration solves for is having the builds be repeatable, consistent, and available. In software we strive for practices to be repeatable; having an externalized build allows for this, which in turn bubbles into consistency. Modern Continuous Integration platforms allow for scaling of the builds (having the builds available when needed vs. having your local machine pegged).

**Best Practices**

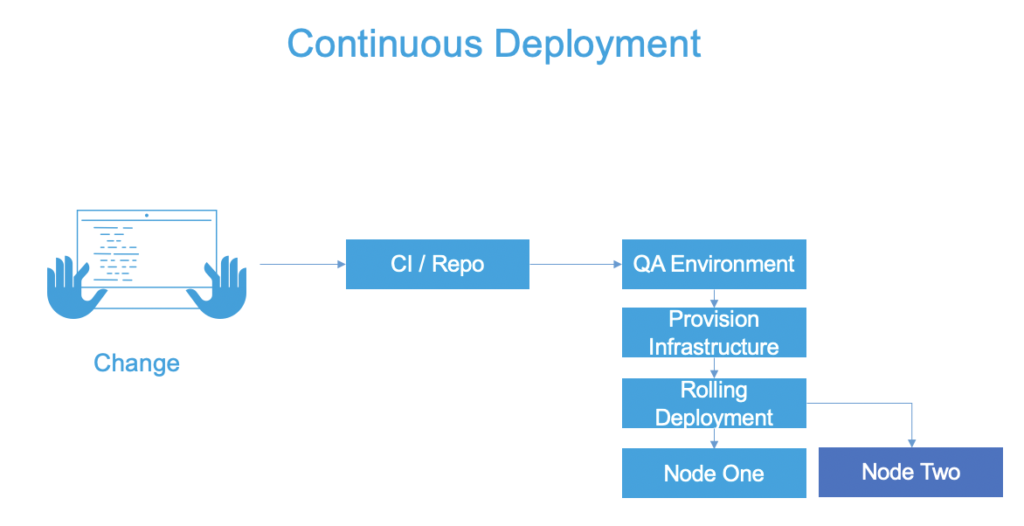
With Continuous Integration, keeping the automated builds fast is key. As this process will be run multiple times a day, with triggers around each commit or merge, time waiting for results can snowball. A challenge observed with Continuous Integration is overstepping into other Continuous pillars, like overburdening Continuous Integration platforms into Continuous Delivery.

Confidence in the build and deployable packaging is different than confidence in the deployment and subsequent release. Being strategic in where to apply parts of your test suite is needed in order to avoid overburdening the Continuous Integration process. A line in the sand should be that Continuous Integration tackles artifact-centric confidence-building (for example, unit tests and artifact scans). Tests that take into account other artifacts and dependencies and/or application infrastructure are better-suited for the Continuous Delivery process. After the build is checked into a central repository, the next step to getting your idea into production is the promotion/deployment process.

**Continuous Deployment**

Continuous Deployment focuses – just like the name implies – on the deployment; the actual installation and distribution of the bits. During a deployment, the application binary/packaging can transverse the topology on where the application or application infrastructure needs to serve traffic. In the traditional sense, Continuous Deployment focuses on the automation to deploy across environments or clusters. As you traverse environments from non-prod to the staging environment and eventually to production, the number of endpoints you deploy to increases. Continuous Deployment focuses on the path of least resistance to get the software into the needed environment(s).

With modern platforms such as Kubernetes, the separation of environments might not be physical when compared to legacy or traditional machine-based platforms. A namespace (software separation) might be all that is separating development from production, though good distributed systems principles still apply regardless of what platform you chose. With distributed systems, the topologies that changes need to propagate to can be large, even in pre-production environments.



**Why it’s Important**

The days of SCPing into boxes and dropping off binary distributions are almost faded away. As Continuous Integration provides a deployable artifact, Continuous Deployment can take that artifact forward into additional environments. Low hanging fruit might be as soon as a new artifact is created, immediately deploy that artifact into a dev-integration and/or quality assurance environment to start integration testing. Especially with modern paradigms, such as immutable applications and infrastructure meaning any change is rebuilt, the number of deployments will increase significantly.

**Best Practices**

Deployments encompass two pairs: the installation/activation pair and the uninstallation/deactivation pair. From a pure deployment standpoint, leveraging a rolling deployment is the defacto standard. A rolling deployment allows for old application nodes to be replaced in an incremental interval, typically one by one, until all the nodes are the new version. The application instance/node being upgraded is taken out of the load balancer pool, then when the installation is complete, it is reconstituted back into the pool.

Having a clear map of the topology, especially if the infrastructure is elastic or on-demand, is key to understanding where your artifacts are going. Similar to the goals of Continuous Integration, keeping the deployment fast is a good goal to have. The appearance of speed can be there if certain tasks have to be run in parallel (ie: spinning up the infrastructure for artifacts to be deployed onto).

Understand that Continuous Deployment and Continuous Delivery have slightly different goals. Continuous Delivery has overlap with Continuous Deployment on the deployment front, but we need to be thoughtful not to overburden deployment systems, confidence-building, safety – which are crucial for Continuous Delivery.

**Task 4**

**Mentoring tools?**

Whether you are just developing your workplace mentoring program or are looking at a revamp, there are some mentoring tools that can help you be more successful. These include assessment and goal-setting tools, technology that can improve communication and connectivity as well as forms for participants to offer their feedback and input.

SWOT Assessment: The very first thing that a mentor and mentee should do when they meet is a SWOT assessment of the mentee. This stands for strength, weaknesses, opportunities and threats. By assessing the strengths and weaknesses of the mentee, goal-setting will become clearer. In the same way, identifying opportunities for growth and threats to the mentee’s development can help define the steps needed to be taken to achieve the goals.

SMART goal-setting: One of the key elements of a successful mentor-mentee relationship is goal-setting. It is also one of the things that set mentorships apart from other workplace relationships. A mentor should guide their mentee to create goals that fit the SMART technique. This means a goal should be specific, measurable, attainable, realistic and time sensitive. It is important that the mentee write down their goals and check to make sure it fits the SMART approach. SMART goals are ones that are easy to track and measure so the mentor and mentee can see the benefits of their mentorship.

Application forms: Although it is not always seen as a tool for mentorship, the application that participants fill in can provide a lot of useful information. These details are particularly important when it comes to the matching process. Mentoring is built around the relationship between the mentor and mentee. That means getting the right match is one of the best ways to ensure success for the participants and the mentoring program as a whole. Workplace mentoring program managers should assess the information they need or want to have from their applicants and ensure those questions are on the application form. However, keep in mind that the application process needs to be balanced. That is, if you are too invasive or if the application process is too time-consuming, you may find you are short of participants.

Resources for a successful mentoring relationship: Mentoring can be a new experience for some mentors and mentees. Administrators can help them have a blueprint for their meetings by providing resources like agendas, handbooks, helpful articles, and questions they can ask one another. These resources can help fill in the gaps when a mentor or mentee isn't sure what to ask or what they should expect from a mentoring relationship.

Feedback forms: Following a mentorship, it is important to get the feedback from both the mentor and the mentee about their experience. This can be done with simple questionnaires or forms, including online survey-style forms. The more specific the questions the better information you can solicit from participants. Feedback is a valuable tool for assessing a workplace mentoring program. It can help managers determine if there are some elements of the program that need to be reworked or changed. Participant reports can also be useful in gauging the success of the mentorship program.

**Task 5**

**What is the importance of differential in training?**

Integration is the answer to removing human error, improving efficiency, and cutting costs.