DevOps - Module 1: Fundamentals of Tech

DevOps

Module 1: Fundamentals of Tech

Video: Foundations of technology

A comprehensive guide for it.
Beginners, I am Ravi, your course instructor.
In this course, I'm going to help you understand the need
of foundations, which are required for anyone who wants
to make their career in it.
Say you're enrolling for any of the IT courses, like
development related courses for front end, which includes
H-T-M-L-C-S-S, JavaScript
and backend related courses like Python, Java,
or platform related courses like AWS, Azure
and DevOps, and also latest internet courses
like artificial intelligence,
machine learning, and deep learning.
To learn these courses, you need to know few
basic software essentials, which are must
for any IT professional and will.
That's what we are going to cover as a part of this course.

Video: What is Application

Well, in simple terms,
an application is a software program, which is designed
to perform a specific task.
For instance, if you take Uber application,
it'll help you book a ride, isn't it?

And similarly, if I take few more examples
of popular applications like Google Maps,
Netflix, and Amazon.
Now if you talk about any of these applications,
they perform some task, isn't it?
For instance, if you take Google Maps, it provides
real time navigation, location services,
and helps you find destinations.
Similarly, if you take instance of Netflix application, it
helps you stream movies, TV shows,
and original content for entertainment purpose.
If you go with Amazon application, it helps you with
online marketplace buying the products
and selling the products.
So with these examples, we can confirm
any application we use, like Google Maps, for instance.
Netflix and Amazon,
all these applications are doing some task.

Video: Types of Application

Applications.
We can broadly divide the applications into
the following kinds.
One is desktop applications.
Another one is web applications followed
by mobile applications and enterprise applications.
Now, what are these different types,
how they will change from one another?
Let's start briefly exploring them.
Let's start with desktop applications.
Applications that are installed
on your computer operating system, which are optimized

for local processing and used offline.
If I take few examples, Microsoft PowerPoint,
VLC, media Player, and Chrome browser.
All these applications we will install in our laptops are
personal computers, and we are going to start using them.
Let's talk about web applications now.
Applications which are installed on a server,
and these applications are accessed
with a web browser generally,
and these applications require internet connectivity.
Few examples include Netflix for watching
and streaming the movies, Gmail and Amazon.
Similarly, if I talk about mobile applications,
applications designed for smartphones
installed from app stores,
these applications often use device features
such as camera and GPS.
If I take example of WhatsApp,
it might use the camera module
to help you make a video call.
Similarly, if I take Uber application
in this particular scenario,
it'll use the GPS module in my system,
and then it'll help me going with the part
and start working with.
If I take Instagram as a use case, we go ahead
and use it for distributing our images.
If I talk about enterprise application,
these are large scale softwares,
helps in managing vast data
And integrating across different departments.
If I take examples, QuickBooks,
a popular accounting software, Salesforce
for customer management and S-A-P-E-R-P solutions.

And if I want to build any of these applications,
we need to know something called
as application architectures.

Video: Web app technologies

Architecture.
A three tier architecture is a widely used
architectural pattern.
It divides an application into three interconnected
layers called as sprinting,
backend, and database.
We'll go ahead and discuss about individual layers
first, let's start with front end layer.
The front end layer is also called as presentation layer.
The focus is going to be on the client side logic,
and the front end is responsible
for displaying the user interface.
Generally, the visual elements we see in web pages
like buttons, menus, logos, et cetera,
all of these are part of print end.
Moving to backend,
backend layer is also called as application layer.
The focus is on the server side.
Logic backend is responsible
for processing the user inputs.
Finally, we'll move into the database layer.
This is also called as data layer.
It is responsible for storing your customer data
and also retrieving the data, which is going to be used
by the application.
And if I take most popular applications, we have
similar architecture to go with.

I have given you an presentation over here
which shows the most popular applications.
For instance, take Google.
The most widely used search engine in the world
is having a architecture.
You can see frontend client side with the help of JavaScript
and TypeScript and going
with the backend technologies.
From the server side, we are using multiple
languages over here like CC plus plus, Golan, Java,
Python node,
and coming to the databases for storing
and retrieving the data.
And to perform data manipulations.
We are using databases like Big Table
And MariaDB.
Similarly, if you take Facebook the most popular
social networking site,
it's also having a three tier architecture.
And similarly, the front end is also JavaScript
and TypeScript coming to the backend languages.
We have Python, c plus plus, Java, et cetera.
And also when it comes to the database, you got MariaDB,
Maya, skill Cassandra, and other databases.
Same goes with YouTube as well,
which is the most popular video sharing site.
It also has the same three tier architecture
with front end being JavaScript and type Script
and backend being Python, Java, cc plus plus Go,
and database being big table MariaDB and et.
I.

Video: CRM Application Use Case

We do have multiple courses
like H-T-M-L-C-S-S, JavaScript, Python,
Java, DevOps, cloud AI, and Machine Learning part.
Now these courses can be growth together
on a broader term,
and these groups, we generally call them as stacks,
like full stack
platform stack and AI stack.
So these combinations can help you build any sort
of application to go with.
As I said, in full stack,
we do have courses like HT, ml, CSS, JavaScript, Python,
and Java In Platform Stack.
We do have courses like DevOps, data engineering
and cloud computing,
and coming to a stack, we have courses like
artificial intelligence, machine learning,
deep learning, and data science.
Now the whole idea is to connect all the stacks together
using an implementation.
And that particular implementation is going to be collection
of all this stacks with an application
to build a CRM application.
Now, what is CRM application?
A customer relationship management, a software tool
that helps the businesses that managing their customers.
Do you know, CRM is used by many companies
for following features, which can help them organize
and manage their customers in a better way.
Key features include lead management,
pre-sales analysis, opportunity filtering,
customers management, sales order vendors,
management, organization structure,
products, inventory, attendance management,

email templates, email and SMS marketing
and calendars.
These features allow businesses
to understand their customers
and provide the insights to the customer behavior
and act accordingly.
Few of the popular CRM applications in the current market
include Salesforce, CRM,
Zoho, CRM, HubSpot,
and yes, there are so many other softwares each in the same
domain, but these are the widely used
now coming to Full Stack team.
Our full stack team in terms of CRM application,
they're the one who are going to build the user interface,
the look and feel of the application,
and as well as the backend logic
which should implement the business
and as well as logical patch.
Now, why full Stack team needs the basics of other stats?
Let's talk from AI perspective to structure the data
for AI features that is providing better AI predictions
and from platform perspective to integrate
with DevOps pipelines for automated builds and deployments.
In the similar way, if we talk from platform perspective
in CRM application, this team is going to build
the infrastructure required so
that we can host the CRM system for global access.
And why do platform stack needs basics
of other stacks from AI perspective
to optimize the cloud cost
and AI workloads, for example,
GPU instances for machine learning
and from full stack perspective
to ensure the data formats match

what stacks and what teams need like JSON for API
to go ahead with the development patch.
And similarly, from a stack perspective,
we are going to build
and train AI models
for our chat bots.
Why a stack needs the basics of other stacks
from full stack perspective to deploy the models
into full stack ui,
which is integrating a chat bot into React.
And from platform perspective
to use DevOps tools
for containerizing the models using Docker.
If we take a real world CRM workflow,
this is a use case where a salesperson
uses the CRM to check a customer's risk
of backing out.
In this context, because of full Stack team,
the risk score appears on the dashboard,
which is built using React.
Because of platform stack data pipelines
fetch the real time customer data from AWS
and Azure environments.
And because of AI stack, machine learning models
can calculate the risk score
using AI agents.
This makes sense to understand how each team
or each stack is giving their insights.
Now, why cross stack basics matter?
If the full stack team does not understand the data formats
of the application, the AI predictions won't display.
If the AI team does not know cloud basics,
there might be a chance of
having the models overloading the service.

If the platform team ignores
full stack needs, the CRM could be close to slow
and very inconsistent.
So that's the reason why you need to understand the pitfalls
of working in silos.
If the focus is on full stack only,
you might have pretty ui, but no AI insights.
And if the focus is on platform stack only,
you might build robust service,
but there might not be any user friendly features.
And if the focus is on AI part only,
you might be able to build smart models,
but there is no way to deploy them properly.
So with this part, we want to give a final pitch
to all the learners who are about
to go with this particular course.
So stick stats are like instruments.
In an orchestra,
you might specialize in violin, like a full stack developer.
You might specialize in drums like a platform engineer,
or you might specialize in piano like an AI engineer,
but still you need to understand the rhythm
and harmony to create a masterpiece.
Same applies for our CRM application.
The CRM isn't built by only coders nor cloud experts
or data scientists.
It's all built by working all of them together
with each other.
So this approach shows
how foundational knowledge is required
and it is going to bridge the gap between different teams
and prevent project failures and unlock innovation.
I.

Video: Software Development Life Cycle

For developing any type of applications,
and that is called AS SDLC,
software Development Lifecycle.
It's a process followed by development teams.
SDLC consists of multiple stages, such as
planning, requirement gathering,
design, coding, testing,
deploying, and maintaining.
Now let's try to understand these different
stages in detail.
Let's first talk about analysis.
In this phase, the project team identifies
and gathers all the requirements for your application.
Then understanding the need of users,
then prepare the features
and functionalities expected from the application,
and looking into any specific constraints
or limitations present within the project.
Now let's talk about the design.
So once the requirements are clear,
the system design phase begins in this phase,
the architecture
and overall structure of the application
is defined.
It also includes designing the database schema,
user interfaces,
and other various components like load balances,
connectivity from web servers to app servers
and apps, servers to database servers.
This will be part of system design.
Now let's talk about implementation.
The implementation phase actually involves the coding

of the application based on the system design
given earlier in here,
software developers are the ones who write the code for each
specific module, ensuring it
aligns with the design
and meets the specified requirements.
As you can see all the code written.
Now, let's talk about testing.
Testing is a crucial phase
to ensure the quality of the application
and different types of testing are performed such
as unit testing, integration testing, and systems testing.
And they're used to identify
and fix bugs and issues.
Now let's talk about deployment stage.
Once the application passes all the testing phases,
and if it is considered stable
and ready for production, it is deployed
to a live environment where the end users
can actually go ahead and access the application.
Once the application is deployed,
we'll talk about maintenance phase
after the application enters the maintenance phase.
Ongoing support is provided.
Any issues are bugs reported by the end, users
are addressed in this particular phase.
And additionally, updates
and enhancements will also be made
to improve the application performance.
Now these phases are common for any kind of application.
We go ahead and work in the software development part.
Now, to apply all of this,
we need some application development methodologies.

Video: Application Development Methodologies

Structured approach to manage
the software development process.
I'll take two different methodologies.
One is called waterfall methodology,
and another one is agile methodology.
Let's try to understand both of these methodologies
and see how they can help in constructing an
application first.
First, we'll begin with waterfall methodology.
It's a linear and a sequential approach
where each phase must be fully completed
before the next phase begins.
And as you can see in the image
provided, we observe a sequential flow
means first we take the requirements, then go ahead
with a design, then development, testing,
deployment, and maintenance.
Now let's see the key characteristics of
waterfall methodology.
The phases are in order, like requirements,
followed by design, followed by implementation, followed
by testing, followed by deployment.
And finally, maintenance.
Fixed scope meaning requirements are
defined upfront and they're rarely changed.
Documentation driven where we have detailed
documentation at every stage.
Rigid structure, limited flexibility
for changes once the project starts up.
So these are the characteristics of waterfall methodology.
Now let's discuss the pros of waterfall methodology.
Simple to understand

and manage clear
milestones and deliverables.
Ideal for projects with stable
and well-defined requirements
and projects with more predictability.
Now let's discuss the cons
of waterfall methodology.
No room for feedback until the testing phase is done.
High risk of delays if requirements change
testing occurs late,
which increases defect resolution cost.
Now let's take an application
where waterfall methodology is more suited.
We will build a payroll system
with fixed government regulations
where requirements are stable
and not often changed once fixed.
So a payroll system would be a better choice of application
per waterfall methodology.

Video: Agile Methodology

And flexible approach
that emphasizes on collaboration
and customer feedback.
The following are the core principles
of agile methodology, also called as
Agile manifesto.
Here we focus on individuals
and interactions.
Rather than focusing on processes
and tools, the emphasis is on working
software over comprehensive

documentation, more customer collaboration
over contract negotiation,
responding to changes instead
of following a fixed plan.
The key characteristics
of agile methodology,
iterative development, which delivers the work
in small functional increments called as sprints.
Continuous feedback, regular
input from stakeholders,
adaptability, graceful towards changing requirements,
and cross-functional teams where developers,
testers, and business analysts collaborate daily.
Now let's discuss the pros of agile methodology.
Faster delivery of usable features,
frequently going with updates, improved
customer satisfaction,
and reduced risk of project failures.
These are all the good things about agile methodology.
Now let's discuss the cons of agile methodology.
In here, we require active customer involvement.
We have less predictable timelines
and also less predictable budgets,
and also lack of long-term planning.
So these are the cons of agile methodology.
Now, let's take an application where
agile methodology is more suited for
developing a mobile application where
user needs evolve rapidly with often
changing requirements.
So developing a mobile application would require
a agile methodology approach to build a software.
With this, we also need to understand about
agile frameworks.

Agile frameworks are basically structured practices
that operationalize agile principles,
offering specific workflows, roles,
and tools to manage the
projects flexibly.
The popular Agile frameworks are
Scrum and Kanban.
Left one is Scrum.
The right one is Kanban, which we should
discuss in detail.

Video: Scrum

In Scrum, we have sprints.
Sprints are time boxed iterations,
which can be in between one to four weeks.
In here, we deliver a working product increment.
Following our scrum roles, we have
scrum master product owner
and development team.
A scrum master facilitates the process
and removes the blockers within your project,
whereas a product owner manages
product backlogs and he will prioritize the tasks.
Coming to development team,
it's a self-organizing group that delivers the work.
Similarly, we do have scrum artifacts,
which are product backlog, sprint backlog,
and product increment.
A product backlog is a prioritized list of features.
A sprint backlog are the tasks selected
for the current sprint product
increment a shippable product version

after each sprint.
Let's also discuss about scrum events,
where we have sprint planning,
daily standup,
or also called as daily scrum
sprint review and sprint retrospective.
First, let's understand what is sprint planning.
A team defines sprint goals
and select the tasks from the product backlog.
A daily standup is a 15 minute sync up
to discuss the progress and also obstacles present.
And we have sprint review
stakeholder demo of the sprint deliverables
to gather the feedback.
Sprint retrospective teams review
sprint process to identify improvements.
Let's discuss about Kanban.
Now, in Kanban,
we use both, which will have visual
workflows like in to do in progress
that is doing and done to track your tasks.
Work in progress limits
restricts multitasking to improve focus,
and we have continuous delivery.
No fixed hydrations tasks
flow continuously.
Now let's summarize
the scrum framework by taking all the roles
like artifacts even
and also scrum roads with respect
to A CRM project
and login module functionality.
In here we see different things we discussed in agile
with respect to this particular framework.

We got scrum master, product owner
development team, product backlog, sprint backlog,
daily scrum sprint review, and sprint retrospective.
And finally, the output.
Let's see about each one in detail.
First, let's talk about scrum Master.
Scrum master ensures the team follows agile
practices and removes blockers.
Next, let's discuss about product owner.
Product owner manages the product backlog
and prioritizes tasks.
Defining a login requirement.
For example, we want to design a login page,
which must support two factor authentication.
In that regard. When we talk about product backlog,
a prioritized list of features.
For example, add a secure user login
to CRM is going
To to be a prioritized user story.
Now let's talk about sprint planning.
Team defines sprint goals
and selects the tasks from the product backlog.
Coming to sprint backlog, the tasks
selected for the current sprint, which are going
to be picked up by development team,
development team, commits to coding the login feature
in a two week sprint.
That is building the login user interface
backend authentication and testing.
And for this, a daily scrum meet will happen,
which is a 15 minute sync up to discuss our
progress and obstacles.
Example, login UI is completed.
Now working on authentication.

Next we got sprint review, a stakeholder demo
of sprint deliverables to gather feedback.
Here it is nothing
but giving a demo
of the working login page to your stakeholders.
Then we have sprint retrospective.
The team reviews sprint process
to identify improvements.
Example, speeding up API integration
in the next sprint, and then final output,
a shippable login feature with email,
password authentication, error handling,
and security checks.
Now say we have
completed the entire CRM application in this manner
so that customers can start accessing the
application features.
Now.

Video: Data Fundamentals

CRM application say, we have completed
the development of CRM application
and we want to host the system so that
customers can start accessing the application.
Now, in this context, I'm going to help you understand
how data and compute come into the picture.
Now, let's connect data fundamentals
to our CRM application.
Let's discuss what is data First.
Data is collection of facts.
Figures are symbols that can be processed
and analyzed to extract useful information.

Now let's see how to categorize the data.
We have structured data,
semi-structured data,
and unstructured data.
These images will help you understand
how there is some sort of connectivity
between the data in the first and coming to the second.
We are missing something and coming to the third.
It is completely different.
We'll go ahead and understand about each type in detail.
Let's talk about structured data first.
Data is organized in a predefined format
in row and columns.
Examples include databases, spreadsheets, and CSV files.
In the context of CRM application,
our customer details will be stored in
SQL database
and sales transactions can be stored in
spreadsheets and CSVs.
Now let's talk about
semi-structured data here.
There is some organization in the data,
but it is not as rigid as structured data.
Examples include J-S-O-N-X-M-L log
files, et cetera.
In the context of CRM
application, we can see
The logs of user activity
are vendor contracts.
Now, let's talk about unstructured data.
There is no predefined format.
There is no structuring in the data.
If I take some examples, like text documents, images,
videos, audio files,

and in the context of CRM application,
email templates can be stored in the PDF format.
Your customer call recordings,
which are audio files in the MP three format,
and your customer images may be in GPG format.
And with this we can say that data can be in the form
of structured, semi-structured and unstructured.
Now, let's try to understand
where this data is stored.
It can be in databases
or it can be a historical data,
which is there in warehouses,
and you might opt for cloud storage.
Now let's talk about each individual of them
and we'll begin with databases first.
Databases are used to store, organize
and re review the data
of your applications efficiently.
Databases can be categorized into SQL
and no SQL databases.
SQL databases can store leads
and customer information in terms of CRM application.
In a relational database like MySQL,
Postgres, et cetera,
no SQL databases can store the vendor's data
in a non-relational database like
no SQL databases, which includes MongoDB,
dynamo db, et cetera.
Coming to data warehouses,
data warehouses are used for storing
large amounts of historical data
for doing analysis and getting
Business insights.
In here, we take data sources from multiple entities

and we go through ETL process
and move the data into warehouses
for reporting analytics and data mining.
Coming to cloud storage,
it is an on demand storage used
to store files, backup, et cetera in the cloud,
which provides high availability and scalability.
Cloud storage options include S3 Blob
and Google Clouds.
This can give you cheap cost storage
and providing the benefits
of high availability and scalability.
Now moving ahead, we have different
techniques for going and managing data.
First, we'll begin with data analysis.
It is used for extracting insights,
identifying trends, and making predictions.
The techniques we use here are basic statistics
and data visualization tools.
Used for performing data analysis are
Excel and business intelligence tools
like Power bi,
and we also have one more stream for managing the data
and advancement called as data engineering.
This is used for building
and maintaining the infrastructure for collecting,
storing, and processing the data.
The key concepts of data engineering are ETL,
extracting, transforming, and loading the data.
We use tools like Spark, Databricks,
and cloud data services from Amazon and Azure.

Video: Cloud Computing

With our CRM application,
as per the earlier discussion we had on data.
Now the question is where this data is stored
and how do we process this data to store
and process the data?
We need storage and compute par.
So in simple terms, it is nothing
but the CPU ram, which is memory
and disc storage for our applications.
Here are the key computing technologies.
You should be aware of CPUA
central processing unit,
which is a general purpose processor
for a wide variety of tasks.
For example, to run CRMs, core features
and user authentication
and transaction processing.
We need CPUs
and coming to GPUs, which is graphics processing unit.
These are specialized for parallel processing,
which is ideal for AI training
and graphic intensive tasks
to train our AI models.
For example, like in the customer relationship management
for lead scoring
and email templates, suggestions, these components are going
to be used and also we have something
called edge computing,
which will be processing the data
closer to the source to reduce latency.
We use Edge Service to
cache the application data in here.
The CRM data will be available
for faster access in regions like Asia

and Europe, where our edge servers are going to be kept.
We can traditionally manage the storage
and compute on
traditional, on-premise data centers,
but the effort time
and capital expenditure is quite high in working
With on-premise data centers.
So we generally opt
for cloud computing nowadays in the modern approach.
Now the question is what is cloud?
Cloud computing is on demand availability
of compute resources.
We discussed earlier like CPU,
ram, hard disk, GPUs, machine learning models,
all these particular entities are available over the
internet first so that you don't need to put lot
of effort, capital,
and time in managing these particular resources.
Now let's understand how cloud offers their
services, and we call them as cloud models.
One is infrastructure.
As a service model IAAS model,
we got platform as a service PAAS model,
and we have software as a service SAS model.
Let's talk about each of these models.
We'll start with infrastructure as service first.
We offer basic computing resources like servers,
storage networks,
and infrastructure as a service is generally used
by infrastructure teams where we host
server storage
and network, a very low level entity.
Now, let's talk about pass model platform
as a service offers operating systems

and middlewares for developing, running
and managing applications.
Past services are generally used by development teams
where we utilize the platforms
to build our own custom applications in a easier way.
Now let's talk about SaaS model.
In software as a service,
we directly provide a software,
which is basically a package software,
which is ready to use.
And these packages can be directly accessed over the
internet by our end users.
So in terms of SaaS directly, we go ahead
and consume the applications rather than building the
platforms are hosting the infrastructure.
Now let's talk about popular cloud platforms,
which are available in the market.
AWS Amazon Web Services from Amazon,
and we got Azure from Microsoft
and we got GCP Google Cloud platform, which is from Google.
And these three are currently the most
in trend in the market.
Now let's talk about cloud computing benefits.
We have reduced costs, we have better security,
we have scalability and easy disaster recovery.
These benefits help the customers
to move from traditional data centers
to cloud data centers, and that's the future.