Task 1: Describe the following

Observability

- The Three pillars of observability
  Logs, metrics, och traces är det man ofta kallar The Three pillars i observability.
  Det är kraftfulla verktyg som har förmågan att bygga upp ett bättre system.
- Blackbox vs. Whitebox monitoring
  Blackbox - övervakning av servrar med fokus på områden som hårddiskutrymme, CPU-användning, minnesanvändning, belastning medelvärde.
  Whitebox - Monitorering av en applikation/process.
- Alerting
  Den tillåter dig att upptäcka problem var som helst i din infrastruktur så att du snabbt kan identifiera deras orsaker och minimera nedbrytning och störning av tjänsterna.
- Monitoring Signals
  Jämförs alltid med inställda mängder för processövervakning för att identifiera processfel på grund av motsvarande betydande avvikelser.
- Why do we need logs?
  De hjälper till att synliggöra hur våra applikationer körs på var och en av de olika infrastruktur komponenterna.
- What is the difference between logs and metrics?
  Logs - Logs handlar om en specifik händelse.
  Metrics - Metrics är en mätning vid en tidpunkt för systemet.
- What is Tracing?
  Innebär en specialiserad användning av loggning för att spela in information om programmets körning.

Service Mesh

- What is a Service Mesh?
  Styr kommunikation mellan tjänster över ett nätverk.
- How does enable it’s data plane?
  Sidecar, som placeras på poddar och kontrollerar trafiken
- mTLS
  Är när flera applikationer delar 1 TLS uppkoppling.
- Circuit Breaking
  Det används för att upptäcka fel och för att förhindra ett fel från att inträffa.
- Traffic splitting/steering
  Traffic splitting - Delar upp trafiken för ett dataflöde över flera access nätverk.
  Traffic steering - Väljer ett åtkomst nätverk för ett nytt dataflöde och överför trafiken för detta dataflöde över det valda accessnätet.
- Fault Injection
  En teknik för att förbättra täckningen av ett test genom att införa fel på testkod, särskilt felhantering, som annars sällan kan följas.
- Rolling Update
  Är konceptet för att ofta leverera uppdateringar till applikationer.

Serverless

- What are some benefits and drawbacks of adopting serverless architecture?
  Benefits - Enklare underhåll, Skalar bättre, Minskar uppstartskostnader.
  Drawbacks - Priser kan gå upp.

Task 2 : Declare how the following solutions solves the following challenges.

Monitoring

- Grafana
  Visar upp data från loggar i ett grafiskt gränssnitt som är enkelt att förstå.
- Prometheus
  Är ett gratis program som används för övervakning och varning av händelser.
- Sentry
  Tjänst som man kan hosta själv eller i molnet som analyserar loggar och upptäcker fel i realtid.
- Kibana
  Är ett verktyg som kan användas för att övervaka system över tid med hjälp av grafer och histogram. Kibana använder sig av ElasticSearch och analysera den datan och upptäcka problem.

Logging

- Fluentd
  Fluent är ett verktyg som kan användas för att skicka loggar till andra servrar, söka igenom dem och fungera mellanlager som kan sköta notiser till ansvariga asynkront.
- Elasticsearch
  Är ett verktyg som kan användas för att söka igenom olika sorters filer.
- Logstash
  Kan samla in och tolka loggar. Loggar öppnas i programmet där man kan identifiera användarna, hämta ut geografisk information från IP adresser och liknande.

Tracing

- Jaeger
  Är en mjukvara för distribuerad spårning, som kan hjälpa till att övervaka och felsöka en komplex miljö för microservices.
- Zipkin
  Hjälper till att samla in tidsinformation som behövs för att felsöka problem i service arkitekturer.
- OpenTracing
  Hjälper utvecklare enkelt att spåra instrument till sin kodbas.

Observability Stacks

- Istio
  Genererar detaljerad telemetri för all service kommunikation inom ett nät.
- Elastic Stack
  Är fyra verktyg ElasticSearch, LogStash, Kibana och Beats. Tillsammans kan de analysera, söka igenom, presentera och transportera loggfiler i format.

Task3 : Choose one of the following topics and write a min 1 page (11 pt, Arial or Times

Continuous Integration/Delivery(Deployment)

Att utveckla och släppa programvara kan vara en komplicerad process, särskilt när applikationer, team och distribution infrastruktur själva växer i svårighet.
Ofta blir utmaningar mer uttalade när projekt växer.
Du vill ha en jämn kvalitet och snabbt kunna leverera ny funktionalitet.

För att utveckla, testa och släppa programvara på ett snabbt och principfast sätt har utvecklare och organisationer skapat tre relaterade men distinkta strategier för att hantera och automatisera dessa processer.

Continuous Integration innebär bara att utvecklarens arbetskopior synkroniseras med en delad huvudlinje flera gånger om dagen.
Sammanfogar all kod från alla utvecklare till en central gren av repo många gånger om dagen och försöker undvika konflikter i koden i framtiden.
Konceptet här är att ha flera devs på ett projekt för att hålla repo huvudgren till den senaste formen av källkoden, så att varje dev kan kolla in eller dra från den senaste koden för att undvika konflikter.

Fördelar:

- Fel hittas och fixas tidigare.
- Att övervaka och mäta systemets hälsa är enklare.
- Öppenhet byggs.

Continuous Delivery beskrivs som den logiska utvecklingen av Continuous Integration alltid kunna sätta en produkt i produktion.
Fokus och nyckel här är att hålla kodbasen i ett distribuerbara tillstånd.
Detta är det vanliga vardags projektet som går ut till allmänheten eller står inför konsumenten.
I dagens värld har du inte råd att släppa ett buggigt projekt, så mindre sprint möjliggör snabbare väntetider för att identifiera buggar och därför snabbare tid att fixa dessa buggar, vilket skapar en mycket mer stabil kodbas tidigt.

Fördelar:

- Lägre risk.
- Funktioner når kunden snabbare.
- Snabbare feedback.

Continuous Deployment beskrivs som det logiska nästa steget efter Continuous Delivery distribuera produkten automatiskt i produktionen när den passerar Quality Assurance.
Den här metoden fungerar bra i företagsmiljöer där du planerar att använda användaren som själva testaren och det kan vara snabbare att släppa.

Fördelar:

- Gör distributioner friktionsfri utan att äventyra säkerheten.
- Förbättra den totala produktiviteten.
- Automatisera de upprepade uppgifterna och fokusera på existerande tester.