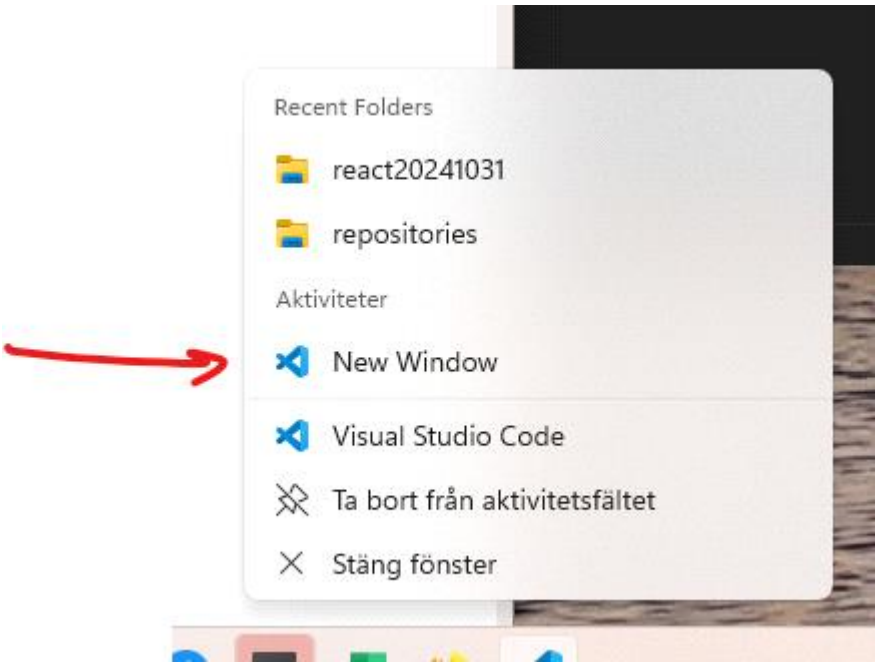
# VS Code som editor

den 31 oktober 2024 15:47

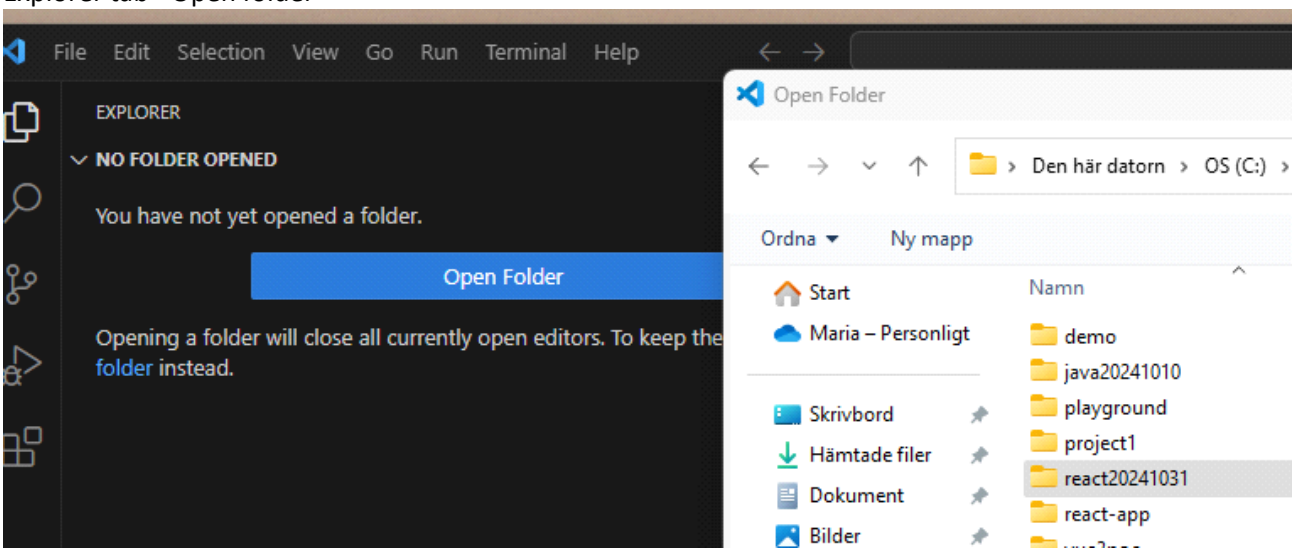
# Öppna projekt på olika sätt

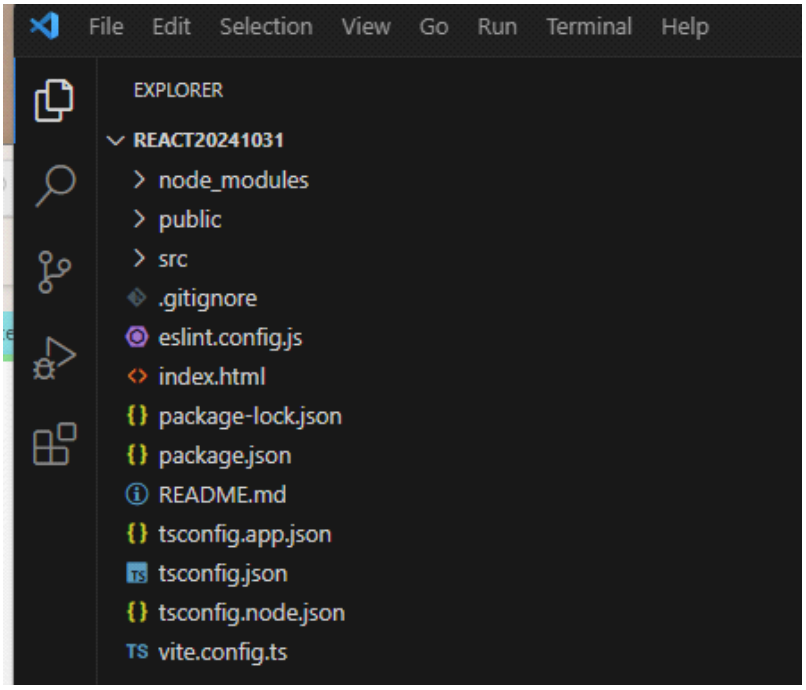
den 31 oktober 2024 15:35

Öppna rätt projekt:
Alt 1: Öppna ett Nytt fönster

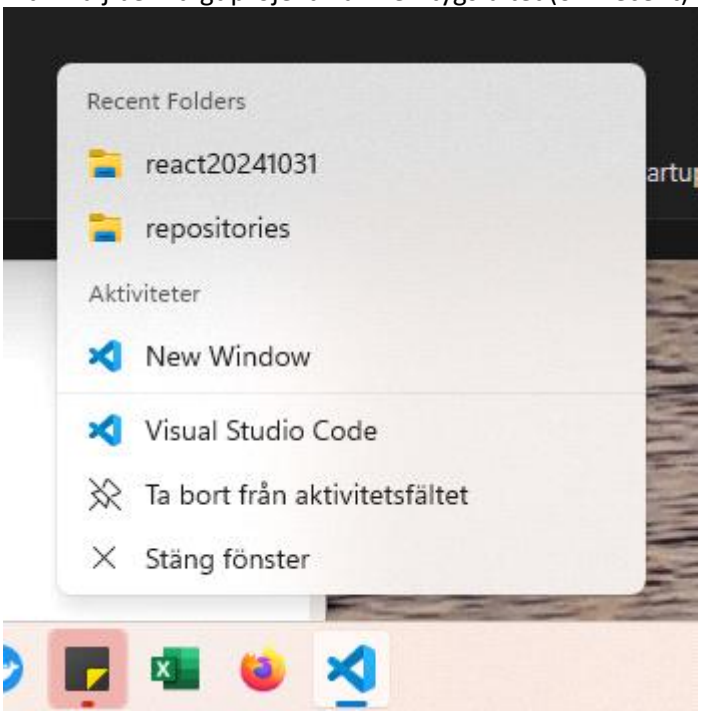



Explorer tab - Open folder

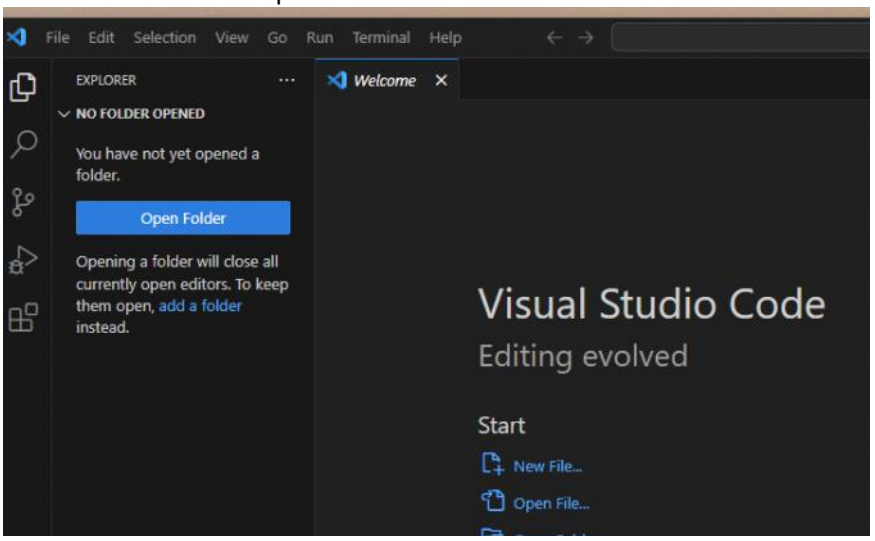


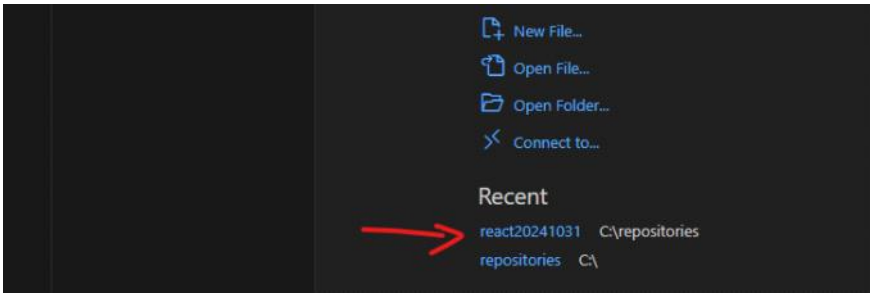



Alt 2 Välj befintligt projekt från verktygsfältet (om recent)




Alt 3 New Window - Explorer Tab - Recent

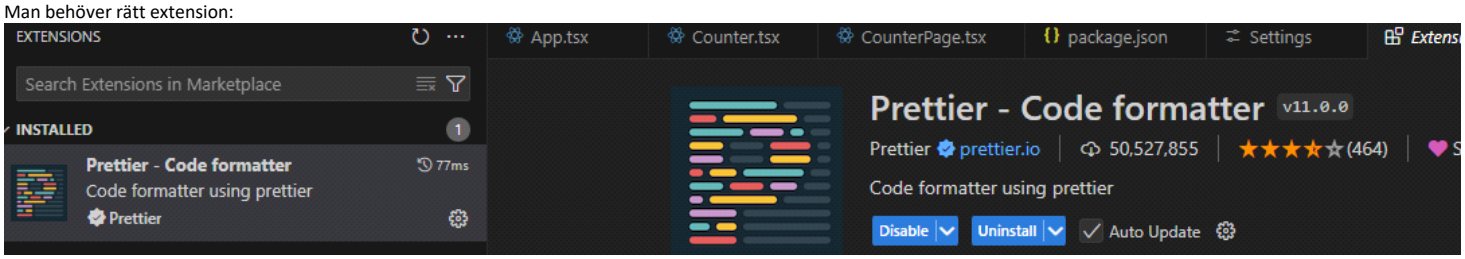



New File...
Open File...
Open Folder...
Connect to...
Recent
react20241031 C:\repositories
repositories C:\

# Prettier

den 1 november 2024 07:06

Man behöver rätt extension:




Dvs inget npm kommando!

Checka i File-Autosave

Därefter måste man göra lite ändringar i settings:

- Sätt default formatter till prettier
- Checka i Format on save
- Ställ in Files: auto delay till något annat än afterDelay. Jag valde onFocusChange

Men funkar detta verkligen nu?

# Steg-för-steg - skapa projekt och börja koda

den 31 oktober 2024 14:58

Tips: För att öppna en terminal i VS code: Välj Terminal - New terminal

Att göra allt en gång från början, via VisualCode:

1. Från extensions-tabben, ladda ner ***prettier*** till VS Code och sätt den "on save".
2. Gå till bas-katalogen, repositories, i terminalen. Skriv **npm create vite@latest**
    a. Gav projektet namnet react20241031
3. Gå till nya katalogen och kör **npm install**
4. Öppna projektet.
5. Kör **npm run dev**
6. Stoppa mha **Ctrl-c**

Andra paket man alltid behöver

- För att lägga till redux toolkit kör **npm install @reduxjs/toolkit react-redux**
    - Detta behöver sedan konfigureras in i main.tsx
    - Skapa en store
    - Skapa valfritt antal slices
- För att använda axios, kör **npm install axios**
- För att få lite snygga csser vid egen lek, kör **npn install bootstrap@latest.** Då behöver man troligen även
    - Kommentera bort allt i App.css
    - I main.tsx, ersätt index.css importen med import 'bootstrap/dist/css/bootstrap.css'

Till browsern behöver man två plugginner:

- React Developer tools https://react.dev/learn/react-developer-tools
- Redux DevTools: https://chromewebstore.google.com/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd

# Editor tips

den 31 oktober 2024 15:57

Autosave: File - Autosave (togglar)

**Command palette:**
**Ctr-Shift-P** + skriv vad du vill göra => Kan snabbt välja detta.
**F1** + skriv vad du vill göra

**Ctr-Shift-P + unused** => Ta bort oanvända importer

**Quick Open shortcut:**
**Ctrl-P** => Listar alla nyligen öppnade filer, kan pila igenom
**Ctrl-P ?** => Får upp tips

**Ctrl-p och skriv filnamn** => Öppna fil

**Andra översikter**
**Ctrl-KS** => Listar alla Keyboard shortcuts
**Högerklick på komponentnamn** => Refactor, anropshieraki mm

**Övriga kommandon**
**Ctrl-shift-f** => Filsök i alla filer
**Ctrl-KC** => Kommentera bort en rad
**Ctrl-KU** => Kommentera fram en rad
**Ctrl-shift-K** => Deleta en rad
**Alt ->** resp **Alt <-** => Navigera mellan nyligen tittade saker

**Högerklicka på komponentnamnet och välj, tex Refactor Move**

- Extract method
- Extract variable
- **F2** => Byta namn på en variabel/klass/...
- Navigera uppåt /neråt i hierarkin

**Rätta till indentering** (Görs via prettier?)

Skapa en run configuration för "npm run dev":???

Debugga:

## Debugger extensions

VS Code has built-in debugging support for the Node.js runtime and can debug JavaScript, TypeScript, or any other language that gets transpiled to JavaScript.

For debugging other languages and runtimes (including PHP, Ruby, Go, C#, Python, C++, PowerShell and many others), look for `Debuggers` extensions in the VS Code Marketplace or select **Install Additional Debuggers** in the top-level Run menu.

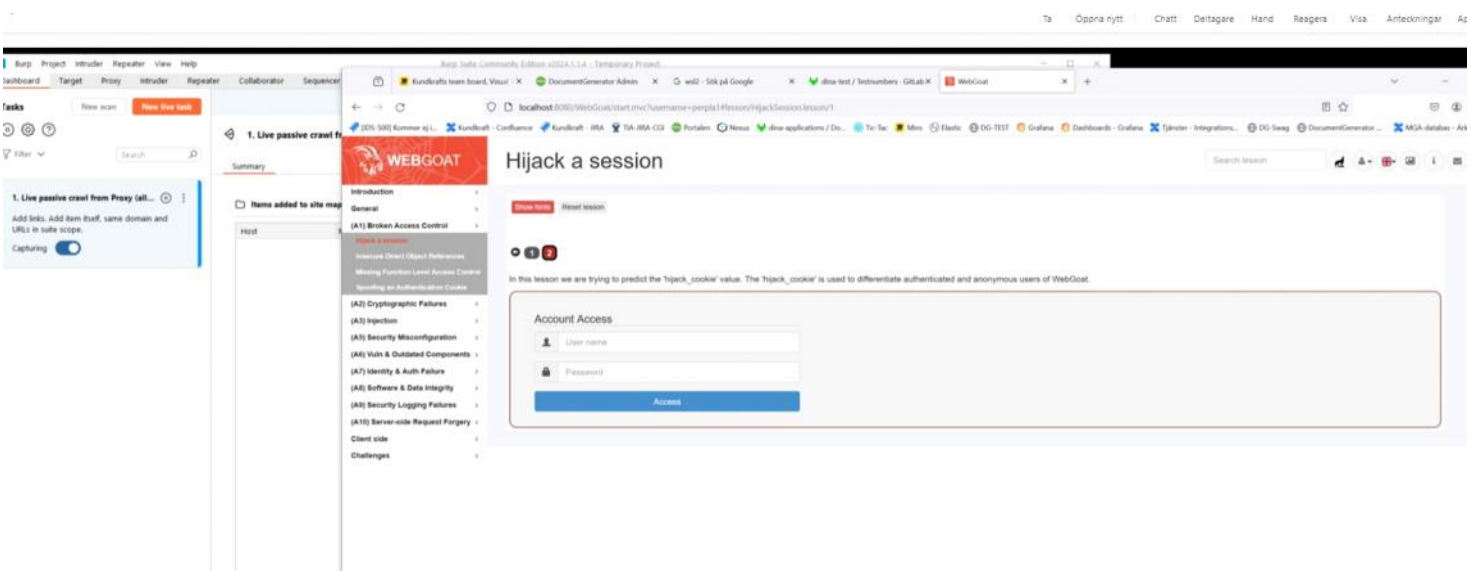
React tutorial:
https://code.visualstudio.com/docs/nodejs/reactjs-tutorial

Se även

- https://code.visualstudio.com/docs/editor/refactoring
- https://code.visualstudio.com/docs/editor/codebasics#_save-auto-save
- Cheet sheat keyboard shortcuts

# Burp mm

den 22 mars 2024 13:21



Kundkrafts team board, Visu | DocumentGenerator Admin | wsl2 - Sök på Google | dina-test / Testnumbers - Gi | WebGoat

localhost:8080/WebGoat/start.mvc?username=perpla1#lesson/VulnerableComponents.lesson

[IDS-500] Kommer ej i... | Kundkraft - Confluence | Kundkraft - JIRA | TIA-JIRA-CGI | Portalen | Nexus | dina-applications / Do... | Tic-Tac | Miro | Elastic

WEBGOAT

- Introduction
- General
- (A1) Broken Access Control
- (A2) Cryptographic Failures
- (A3) Injection
- (A5) Security Misconfiguration
- (A6) Vuln & Outdated Components
- (A7) Identity & Auth Failure
  - Authentication Bypasses
  - Insecure Login
  - JWT tokens
  - Password reset
  - Secure Passwords
- (A8) Software & Data Integrity
- (A9) Security Logging Failures
- (A10) Server-side Request Forgery
- Client side
- Challenges

# Vulnerable Components

Search lesson

Reset lesson

1 2 3 4 5 6 7 8 9 10 11 12 13

## Concept

The way we build software has changed. The open source community is maturing and the availability of open source softv determining the provenance of the libraries used in our applications. Ref: Software Supply Chain

This lesson will walk through the difficulties with managing dependent libraries, the risk of not managing those dependenc are at risk.


Figure: The continued growth of Open Source software.

## Goals

- Gain awareness that the open source consumed is as important as your own custom code.
- Gain awareness of the management, or lack of management, in our open source component consumption.
- Understand the importance of a Bill of Materials in determining open source component risk

# Microservices vs Distributed monolith

den 29 juni 2024 08:57

**Mikrotjänst**

Mikrotjänster kan vara ett väldigt användbart arkitekturmönster, men inte för att lösa integrationsbehov mellan olika applikationer, utan som ett sätt att dela upp monolitiska applikationer i mindre, autonoma delar som underlättar förvaltning och vidareutveckling.

Enkla fristående tjänster som tillsammans bygger upp ett större system.

- En tjänst definieras utifrån sitt rest API
- Små autonoma delar som hanteras helt separat. (Mina sidor / Köpflöden/ Partners/ Cabas/…)
  - Lagom många utvecklare i varje del
  - En ändring i ena applikationen påverkar inte den andra, kan releasas separat
- Tjänsten kan skalas upp vid behov, till flera hårdvaruinstanser.

Det är lätt att blanda ihop begreppen när man pratar om mikrotjänster, webbtjänster, tjänsteorientering o.s.v. Mikrotjänster – trots att de ofta använder samma teknik som webbtjänster – är dock ett ***applikationsmönster*** snarare än ett ***integrationsmönster***.

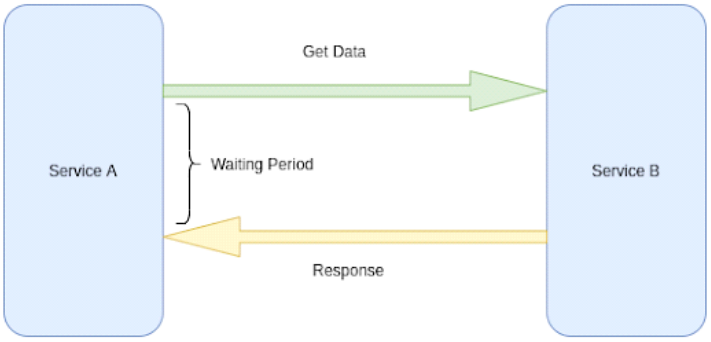
**Distribuerad monolit**

- Vi har microservices arkitektur
- Vi saknar många av kännetecknen av en distribuerad monolit.

Dvs:

- Är det vi gör eg ett integrationsmönster/webbtjänst och inte applikationsmönster/mikrotjänst?
- Vi har en distribuerad monolit pga:

Synchronous Communication




Synchronous communication between two services increases coupling.

You can say service A depends on service B if it either needs data or any data validation from B. It is because both services are in synchronous communication. So if service B goes down or delays the response, it can affect the throughput of service A. if your application has too much synchronous communication between services, it can be a distributed monolith, even if it implements the microservice architecture.

Detta gäller

- Mypages -> ASR, ECM Services, mm
- Framöver om tia-adaptern skulle bli services.

På mina sidor har vi även problem med alla våra andra integrationer (Onbase och Tia) men det är ju inte konstigt, det är ju våra databaser. (För att vi är en webtjänst??)

# Babbens kuberneteskurs 2023-04-12

Sunday, 2 June 2024 22:02

## Agenda

- Snabbkurs om containers
- Vad är Kubernetes?
- Varför Kubernetes?
- Hur man kan köra Kubernetes
- Hur man skapar ett kluster
- Hur man kan rulla ut sin mjukvara
- Övervakning
- Frågor

Paketering, skala upp, slipper att andra applikationers problem påverkar

## Kort om containers

### Varför containers?

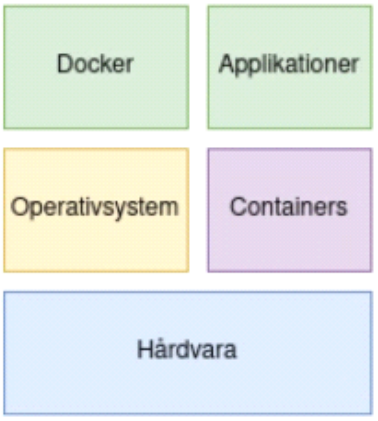
- Paketering
- Standardisering
- Isolering

Virtell maskin är en overhead, containers är lättvikt

# Kort om containers

Vad är en container?

- Lättviktig!

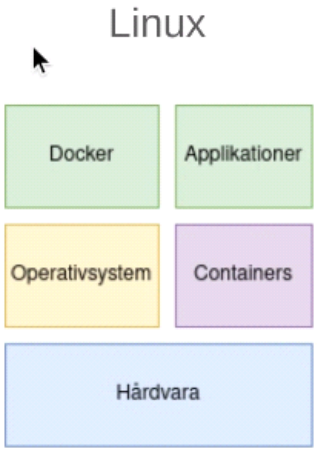



# Kort om containers

Containerplattformar

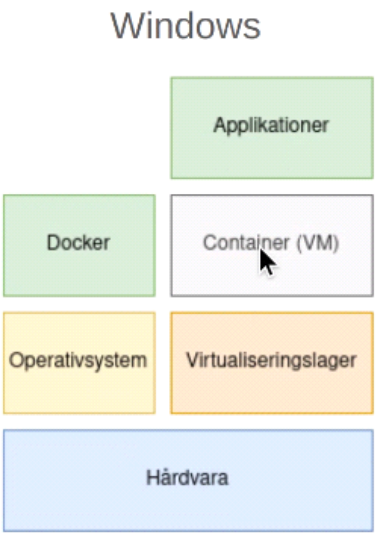
- Docker
- Podman
- LXC
- Containerd

# Containerstöd i olika operativsystem




Native i opeativsystemet ingen overhead

# Containerstöd i olika operativsystem




Så här rä mycket mer overhead – varje kontainer är en virtuell maskin

Man skapar en container från en image
Allt man behöver lägger man upp i ett repository

# Containers

- Paketering: Image
- Tillgängliggöra: Repository
- Bygga vidare: Lager
- Bygga image: Dockerfil
- Filsystem
- Volymer

Filsystem som är skiljt från värdens filsystem. När containern uppgraderas så försvinner filerna
Därför fyller man sån info i volymer

# Verktyg för containers

- Bygga & Managera Docker-klienten (docker)
- Samordna flera containers: Docker-compose
- Hantera maskiner med docker: Docker-machine
- Sätta upp lokalt: Docker-desktop

Docker desktop är för windows bla

# Livet före Docker & Kubernetes

## Vad är Kubernetes

- Plattform för att köra applikationer
- Kluster
- Abstraktion

Skalbara om man behöver fler/färre kluster

Man kan sätta etiketter på maskiner för att kunna köra vissa applikationer på vissa maskiner

## Varför Kubernetes

- Orkestrera hård- och mjukvara
- Lastbalansering
- Skalning
- Redundans
- Effektivt resursutnyttjande

Applikationen finns på flera instanser, dvs även om en maskin går ner så funger

# Varför inte Kubernetes

- Enstaka instanser av applikationer
- Monoliter

Då är det overkill, och opraktiskt.
Passar för microtjänster

# Hur man kan köra Kubernetes

- För utveckling & test: Lokalt på din dator
  - Microk8s (Ubuntu)
  - Minikube
  - K3s
- Köpa som tjänst
  - Kubernetes as a service (KaaS)
  - Finns hos alla stora molnleverantörer
- I molnet
  - Publikt eller privat
  - På virtuella maskiner
- Fysisk hårdvara
  - Metal

Minikube i windows

# Hur man skapar ett kluster

- Ansible: Kubespray
- Färdiga images
- Bygga sin egen image

## Hur man kan rulla ut sin mjukvara

- Templates
- Helm
- ArgoCD
- FluxCD

## Övervakning

- Metrics
  - Prometheus
- Loggar
  - Samla in loggar
    - Promtail
    - Logstash
    - Fluent

## Övervakning

- Probes
  - Health
  - Readiness

Health, readiness (kan ta ett tag att starta, är ju fortfarande inte unhealthy

# Övervakning

- Probes
  - Health
  - Readiness
- ELK-stacken
- Grafana
- Alarmhandler

ELK: Elastic search, logstash, kibana

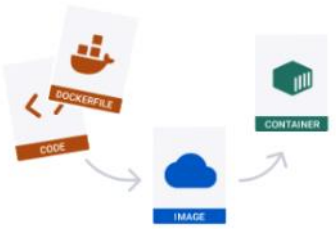
Inga loggar sparas I logstash, skickas, istället för att spara till logfil så ingen rullning etc.
Databas i kubernetes eller inte? Postgres passar till kubernetes.
Stateless workloads är vad kubernetes är byggt för, dvs statefullt är en efterkonstruktion.

För att lära sig:
Kör lokalt på egen dator.

# Docker

den 20 juni 2024 07:01

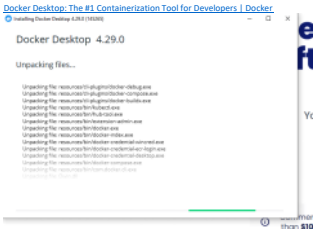
Länkar att titta på
https://docs.gitlab.com/ee/user/packages/container_registry/build_and_push_images.html

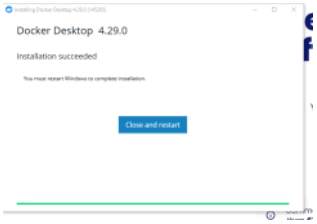


# Docker - intro

den 24 april 2024 15:09

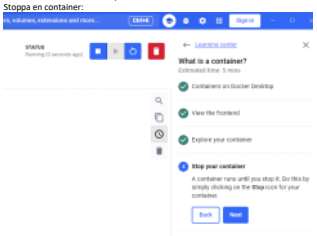
Docker Desktop: The #1 Containerization Tool for Developers | Docker



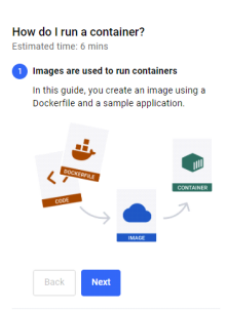




Klicka på porten så kommer man till frontenden.
Klicka på containern så kommer man till den:

Finns flera flikar här att kolla på!
Stoppa en container:

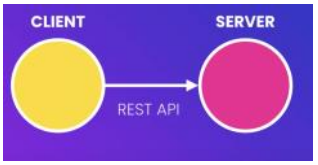


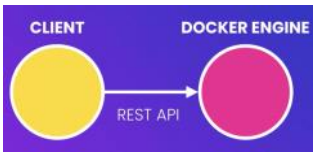
Kan starta och stoppa här!

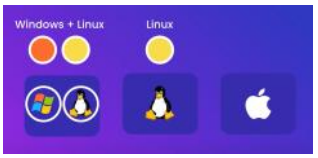



A Docker Tutorial for Beginners (docker-curriculum.com)


Docker Tutorial for Beginners - YouTube

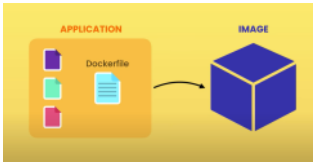



På Linux kan man bara köra Linux containers, men på windows finns både windows och Linux kärnor, vilket innebär att man kan köra även linux containers på windows.

Hur få applikationer till docker?
Skapa en dockerfile, som har all info för att paketera applikationen som en image. Imagen innehåller allt som applikationen behöver för att köra.

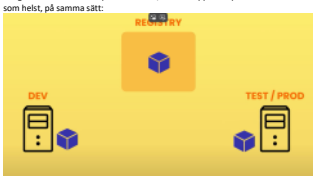


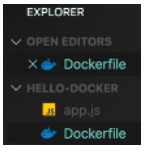


Därefter:
Säg till docker att starta en container med denna image

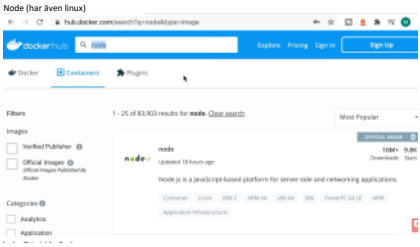
En container är en process, med eget filsystem, som tillhandahålls av imagen.

Pusha docker imagen till docker-hub (som github). Dvs en storage för docker images. När den väl finns på docker-hub, vi kan stoppa den på vilken maskin som helst, på samma sätt:

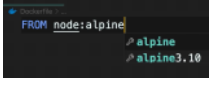



Det finns ett antal images att utgå ifrån på hub.docker.com, tex
Linux
Node (har även linux)

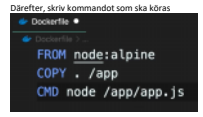


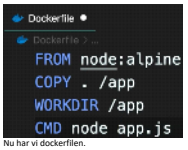
Det finns flera node images, byggda på olika distribueringar av linux



Börja med en sådan image, därefter ladda ner allt från vår applikation till filstrukturen i denna image, skapa katalogen app där

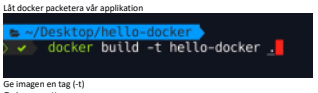
Därefter, skriv kommandot som ska köras




Nu har vi dockerfilen.

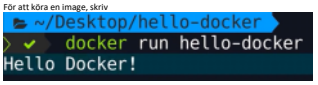
Låt docker packetera vår applikation




Ge imagen en tag (-t)
Ge imagen ett namn
Docker filen finns "här".

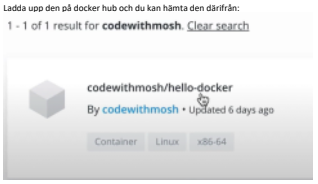
För att hitta alla docker images på den här maskinen:
Skriv docker images
Eller docker image ls


För att köra en image, skriv




Det spelar ingen roll var du står, imagen innehåller allt som behövs.

Ladda upp den på docker hub och du kan hämta den därifrån:

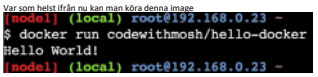



Han gick till en lekyta- play with docker, där man kan starta en ny VM med docker i och köra därifrån.

För att hämta imagen skriv


=> finns här:

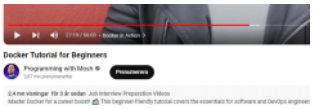

Var som helst ifrån nu kan man köra denna image




Dvs:
Till varje applikation, kan man lägga till en dockerfile.

Därefter kan man skapa en image och köra den på vilken maskin som helst som har docker.

Fortsätt här:
Docker Tutorial for Beginners (youtube.com)



# Docker - grundkurs / splunk intro

den 20 maj 2024 09:32

**Översikt**
Ha dockerdesktop installerat.

**Dockerfilen**
1. Linux - alpine
2. Java - peka ut tex mypages applikationen
3. Maven - starta applikationen via typ mvn spring-boot:run -Dspring.profiles.active=production

```
Dockerfile •
FROM node:alpine
COPY . /app
WORKDIR /app
CMD node app.js
```

När man bygger en gång kallar denna för image, och det är den man kan köra i en container.

Man behöver alltså inte längre ladda ner programmen separat, man kan köra dem i docker, tex splunk.

**Splunk**
Inuti docker kör applikationen splunk
(Man skulle kunna köra splunk på vanligt sätt, men då måste man rensa indexeringen efter varje gång, eftersom den finns kvar annars.)

Om man skulle köra mypages i docker behöver man ha en instans mina sidor, en third-party, mm.

**Docker compose**
Startar alla ingående applikationer samtidigt och kan kommunicera i samma nätverkskontext.
Docker compose kommandon är mycket enklare än docker, kör på den.

Man behöver då en docker-compose.yml fil.

Här är ett ex på docker-compose.yml
```
version: '3'
services:
splunk:
image: splunk/splunk:latest
container_name: splunk
env_file: .env
ports:
- "9001:8000"
- "8088:8088"
volumes:
- ./log_directory:/var/log_directory
```

(Ovan har docker själv fyllt i en image.)

Är ekvivalent med docker
```
docker run -d -p 8000:8000 -e "SPLUNK_START_ARGS=--accept-license" -e "SPLUNK_PASSWORD=<password>" --name splunk splunk/splunk:latest
```

Vad gäller pxw till splunk i .env, hitta på något som är "komplext".
För detaljer, se senare text.

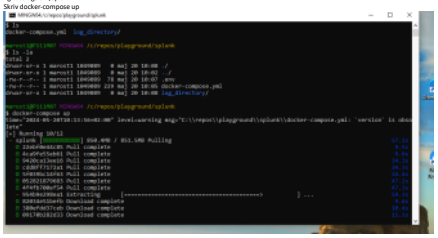
Skapa ett helt tomt intellijworkspace.
För att skapa en docker container för splunk, skapa en katalog med följande innehåll:

- log_directory = Där jag vill att splunk ska indexera filerna.
- Docker-compose.yml ska se ut som ovan
- Ange diverse argument här (eller i docker-compose.yml)

Tidigare kunde man bara köra från terminal, men nu finns en hel del via intellij, tex borde det gå att trycka på pilen i intellij, för att köra docker-compose från intellij (som administrator...). Men det saknas något för att kunna köra med pilen. => Kör istället från gitbash.

(Det borde även fungera via docker desktop. Det gick inte tidigare.)

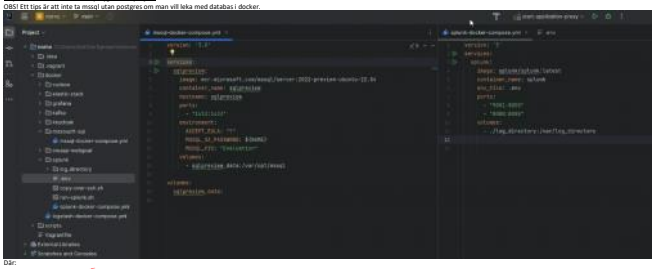
**Köra docker compose från gitbash**
I gitbash gå till /splunk
Skriv docker-compose up



=> Nu finns den containern här:

(Den har bara en container i sig, men kunde förstås haft fler.)
Att stoppa containerns:
Via gitbash, tex stäng gitbash. Eller via docker desktop

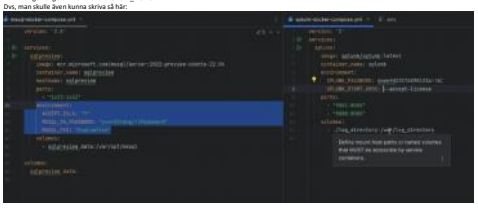
**En på Gidepos struktur i stil "playground" projekt**
OBS! Ett tips är att inte ta mssql utan postgres om man vill leka med databas i docker.



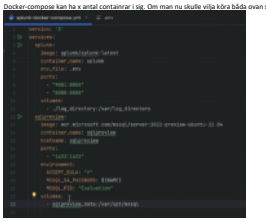
Där:

1 är min maskin, 2 är i containern
Guit är på 9001, på 8088 är något annat....

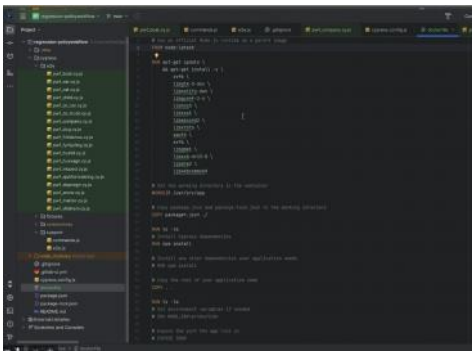
Environment
Kan antingen ange environment: eller env_file: .env.
Dvs, man skulle även kunna skriva så här:



Docker-compose kan ha x antal containrar i sig. Om man nu skulle vilja köra båda ovan samtidigt, klistra in det andra i samma:



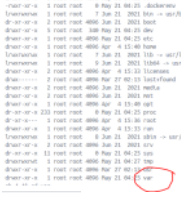
**Annat exempel, att kunna köra policyweblows cypress tester via docker**



Dvs kan köra denna via npm test...
Är:

**För att få ner filerna till log_directory**
Högerklicka på logs i prod och gör download och spara under log_directory
Nu ligger de alltså här:

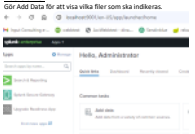
**Docker desktop**
Gå till containern och välj fliken Exec
Här ligger min logg katalog, dvs den är tillgänglig i containern.



Här kommer man alltså åt filerna, se ls -la:

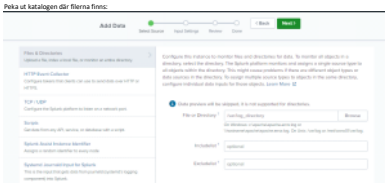
Om man nu stoppar containern och startar om den finns allt kvar, men det är inte självklart, beror på att det är så splunk fu ngerar. Tex för databas, måste man speca volumes! (Dvs det är bra, men då finns å andra sidan databasen kvar...)

**Indexera filer**
Gör Add Data för att visa vilka filer som ska indikeras.



Add data -> Monitor -> Files & directories -> peka ut var/log_directory

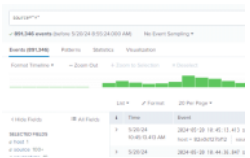
Peka ut katalogen där filerna finns:



Bläddra igenom wizarden med next.

Dvs nu pekar jag ut och har indexerat dem => är sökbara.
När jag stoppas splunk skulle det kunna vara så att man måste indexera om (dvs peka ut katalogen igen), men det verkar inte så. Dock måste man tänka på rangen som kontrolleras så det hittas:

**Splunk**
Gå till localhost:9001 och logga in på splunk:
User: admin
Psw är det jag angett i filen



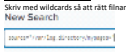
# Splunk grundkurs forts

den 20 maj 2024 15:31

För syntax, testdata och tutorial, se https://docs.splunk.com/Documentation/Splunk/9.2.1/SearchTutorial/Startsearching

Börja med source="*"

**För att välja rätt logzfiler**
Skriv med wildcards så att rätt filnamn pekas ut, tex:




Alla tillgängliga listas:

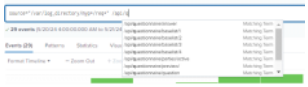


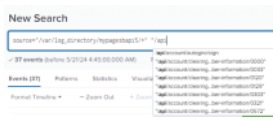

**För att välja rätt sökord**
Klicka på något och välj add to search


Kombinera olika uttryck med mellanslag emellan

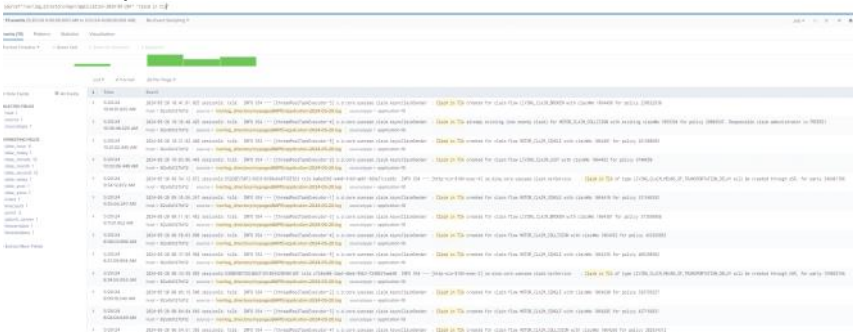
Ex:
source="/var/log_directory/mypages*/requests*" "/api/parties"

Alla som finns i indexeringen dyker upp här:

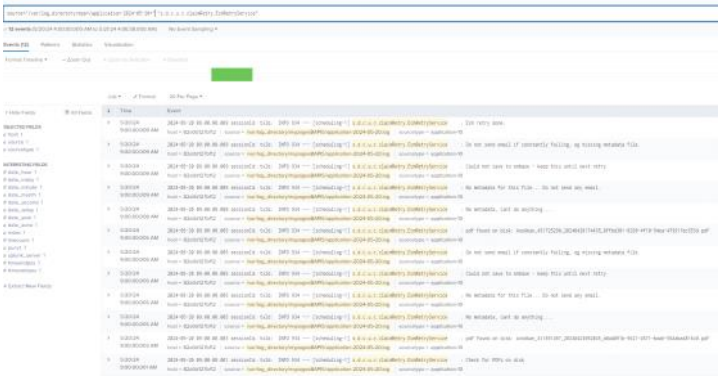




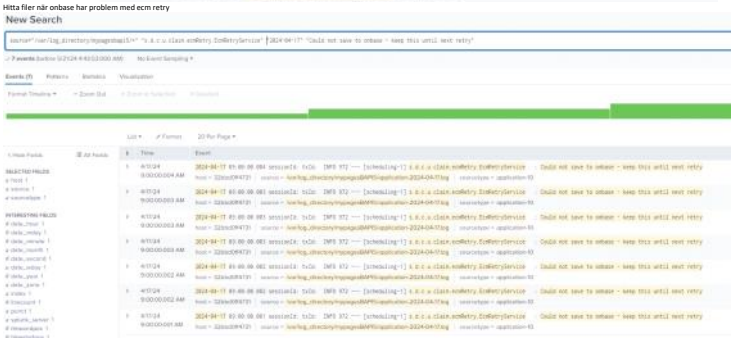

Lite exempel
För att söka fram alla skador som registrerats



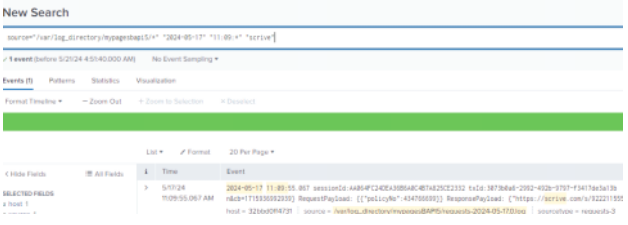
Kolla Ecm-retry



Hitta filer när onbase har problem med ecm retry




Söka fram ett visst scriveid
Questionnaire/sign => datum och klockslag =>



# Splunk - fler exempel

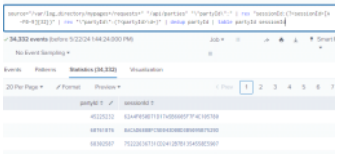
06:17

**Exempel - simpel**
source="/var/log_directory/mypages*/requests*" "/api/parties" "\"partyId\":"
| rex "sessionId:(?<sessionId4>[A-F0-9]{32})"
| dedup sessionId4
| tojson sessionId4
(använd shift+enter och mellanslag för att formattera lite)

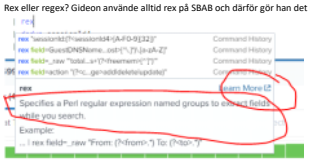
Tänk så här:
Första raden sållar ut så gott det går.
| betyder pipe
Andra raden plockar ut alla "sessionsid:xxxxx" som förekommer och kalla variabeln för något, tex "sessionId4"
Bygg på med att ta bort alla dubletter av sessionsidn
Gör sedan om det till json (eller table eller något annat)

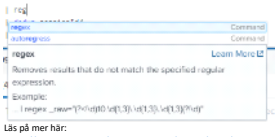
Kan sedan göra mer komplicerat genom att även kolla på partyId (och sedan ta bort duplikat på partyId istället...)

Gideons exempel var egentligen så här:
source="/var/log_directory/mypages*/requests*" "/api/parties" "\"partyId\":" | rex "sessionId:(?<sessionId>[A-F0-9]{32})" | rex "\"partyId\":(?<partyId>\d+)" | dedup partyId | table partyId sessionId



Rex eller regex? Gideon använde alltid rex på SBAB och därför gör han det nu också. Testa båda!!




Läs på mer här:
https://docs.splunk.com/Documentation/Splunk/9.2.1/SearchReference/Regex?ref=hk
https://docs.splunk.com/Documentation/Splunk/9.2.1/SearchReference/Rex?ref=hk

**Ex - komplex 1:**
source="/var/log_directory/*"
(
361AFBF065229E650FAA5677B17A151E OR
0DF1E93A083FFA48FAC6C06F62906076 OR
8AFCEDE9BDBEBDA9512939DDAE0B2932 OR
C511270DC69C7C222A7A900127989BF5 OR
4421A60FA190A9ECA7F55207432F3AA4 OR
DF4C0AF7B7226210162 6E054F4BCA945
)
(
("api/parties" OR "/api/policyForPayer" OR "Called by other than payer or policy holder")
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| rex "\"name\":\"(?<name>[^\"]+)\""
| rex "\"registrationNo\":\"(?<ssn>(\d{12}))\""
| rex "policyForPayer/(?<policyId>(\d+))" | table _time sessionId policyId name ssn _raw

Bygga upp denna själv
source="/var/log_directory/mypages*/*" (B08909A75B2E4F1446AB77D3C933E66D OR
62A4F050D71D17A5B6605F7F4C105780)

source="/var/log_directory/mypages*/*"
(B08909A75B2E4F1446AB77D3C933E66D OR 62A4F050D71D17A5B6605F7F4C105780)
("api/parties" OR "/api/policyForPayer" OR "Called by other than payer or policy holder")
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| rex "\"name\":\"(?<name>[^\"]+)\""
| rex "\"registrationNo\":\"(?<ssn>(\d{12}))\""
| rex "policyForPayer/(?<policyId>(\d+))"
| table _time sessionId policyId name ssn _raw

2024-05-30

**Ex - komplex 2**
## Join example (combining /api/parties with /api/policy-documents-ECM)
source="/var/log_directory/my-pages*/requests*" "policy-document-ECM"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| rex "/api/policy-document-ECM/(?<referenceId>\d+)"
| table sessionId referenceId _time
| join sessionId [
search source="/var/log_directory/my-pages*/requests*" "/api/parties" "\"partyId\":"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| rex "\"partyId\":(?<partyId>\d+)"
| table sessionId partyId
]

(Men denna join fungerar inte...)

# Splunk Ex på kommandon fr Dina

den 17 juni 2024 14:48

Detta ger 33 events:
source="/var/log_directory/*" "No instance of party with registration number"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table sessionId
| join left=L right=R where L.sessionId = R.sessionId [
search source="/var/log_directory/*" /api/parties "\"partyId\":"
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| table sessionId ]

sessionId verkar behövas ör att kunna göra join
- Visa ***även*** partyid i listan
- _time visar inbyggd tidsstämpel

source="/var/log_directory/*" "No instance of party with registration number"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId
| join left=L right=R where L.sessionId = R.sessionId [
search source="/var/log_directory/*" /api/parties "\"partyId\":"
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| table sessionId partyId]

Om man vill få med hela loggen:
- _raw visar hela loggraden

source="/var/log_directory/*" /api/parties "\"partyId\":" F1085D2125EC0C6481628A64B378D194
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup partyId
| table _time sessionId partyId _raw

För att få unika entries i fråga nr 2:
source="/var/log_directory/*" "No instance of party with registration number"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId
| join left=L right=R where L.sessionId = R.sessionId [
search source="/var/log_directory/*" /api/parties "\"partyId\":"
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup partyId
| table sessionId partyId]

Hej Petra,
Jag har tagit fram en bra sökning i Splunk som går att utnyttja och modifiera för liknande sökningar. Det är användbara byggstenar för mina sidor.

Grundtänk
Första raden sållar ut så gott det går.
- | betyder pipe
- Mha rex plockar man ut det man vill
  - Tex de "sessionsid:xxxxx" som förekommer
  - <sessionId> är namnet på variabeln man sedan använder sig av
- Dedup tar bort ev dubletter
- Ev presentation – tex till table eller json

Formattera gärna med shift-enter för att få det läsbart
Man kan joina ihop olika sökningar
Det finns inbyggda funktioner, tex _time och _raw (ger hela loggraden), se exempel nedan.

Supportärendet
Jag har två sökningar som kombineras med varandra, och sedan har jag exporterat denna lista till excel (som en csv fil som excel kan öppna).
1. Hitta alla sessionsid som har detta fel
2. I dessa sessionsid, leta upp partyId

source="/var/log_directory/*" "No instance of party with registration number"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId
| join left=L right=R where L.sessionId = R.sessionId [
search source="/var/log_directory/*" /api/parties "\"partyId\":"
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup partyId
| table sessionId partyId]

Ger resultatet:

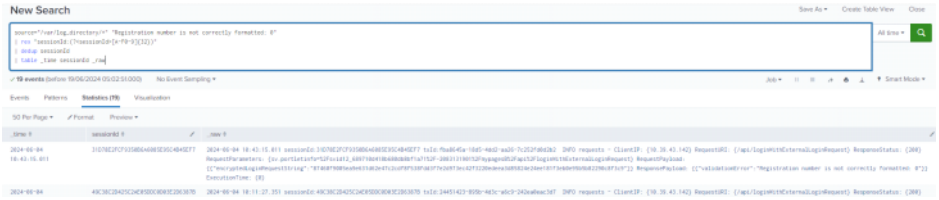
Enskilda sökfrågor, exempel
Sökning 1 ovan – ta fram en lista på alla unika sessionsid bland alla loggarna jag har laddat ner till katalogen var/log_directory
source="/var/log_directory/*" "No instance of party with registration number"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId

Sökning 2 – utifrån ovan sökning, ta fram en lista på partyId för varje sessionsid
source="/var/log_directory/*" /api/parties "\"partyId\":"
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup partyId
| table sessionId partyId

(Om man skulle söka på enbart denna fråga skulle man dock få långt fler svar.)

Annat exempel på syntax
source="/var/log_directory/*" /api/parties "\"partyId\":" 68242428
F1085D2125EC0C6481628A64B378D194
| rex "\"partyId\":(?<partyId>\d+)"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| table sessionId partyId _raw

**Fler bra sökningar**
(Bara att byta ut textsträngen allteftersom)
source="/var/log_directory/*" "Registration number is not correctly formatted: 0"
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId _raw



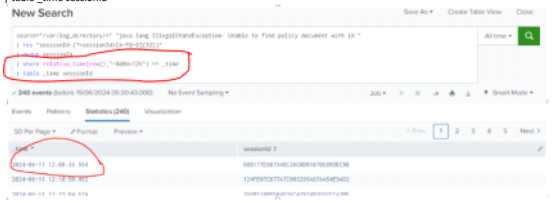

**Begränsa i tiden**
Alt 1 I den inledande sökningen, lägg till earliest och latest
Ex på format:
- earliest=-8d@d+3h latest=@d
- earliest="06/15/2024:20:00:00" latest=now

source="/var/log_directory/*" "java.lang.IllegalStateException: Unable to find policy document with id " earliest=-8d@d+3h latest=@d
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId

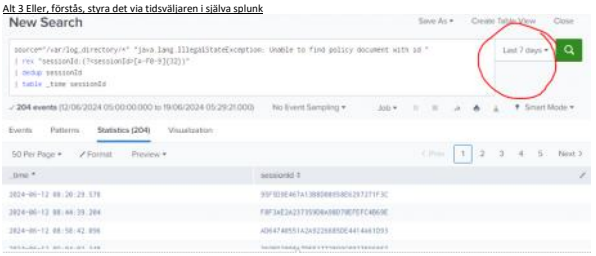
source="/var/log_directory/*" "java.lang.IllegalStateException: Unable to find policy document with id " earliest="06/15/2024:20:00:00" latest=now
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId


Alt 2 Begränsa resultaten i senare skede
Ex på format
- | where relative_time(now(),"-8d@d+12h") <= _time

source="/var/log_directory/*" "java.lang.IllegalStateException: Unable to find policy document with id "
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| where relative_time(now(),"-8d@d+12h") <= _time
| table _time sessionId




Alt 3 Eller, förstås, styra det via tidsväljaren i själva splunk



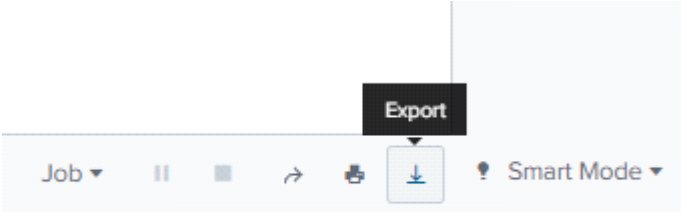

**Hitta policyNo och policyHolderId**
source="/var/log_directory/*" 3751BF0B7CE1F16174A536BE00DD6393 "/api/activePolicy/search"
| rex "\"policyNo\":(?<policyNo>\d+)"
| rex "\"policyHolderId\":(?<policyHolderId>\d+)"
| table policyHolderId policyNo _raw

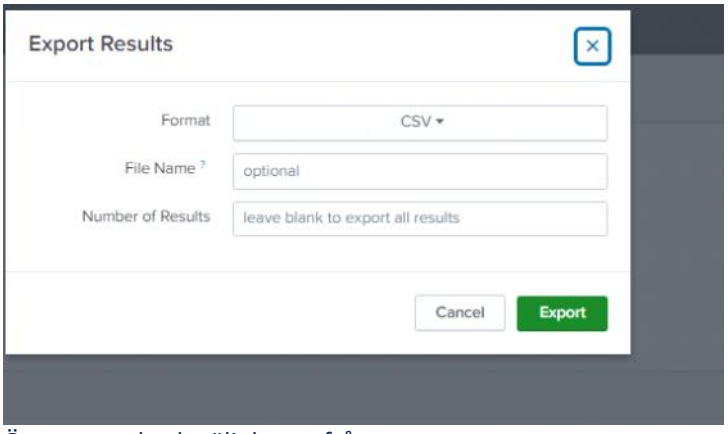
Dvs sammantaget:
source="/var/log_directory/*" "java.lang.IllegalStateException: Unable to find policy document with id " earliest="06/17/2024:13:08:00" latest=now
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| dedup sessionId
| table _time sessionId
| join left=L right=R where L.sessionId = R.sessionId [
search source="/var/log_directory/*" "/api/activePolicy/search"
| rex "\"policyNo\":(?<policyNo>\d+)"
| rex "\"policyHolderId\":(?<policyHolderId>\d+)"
| rex "\"policyType\":\"(?<policyType>[A-O]*)\""
| rex "sessionId:(?<sessionId>[A-F0-9]{32})"
| table sessionId policyHolderId policyNo policyType
]

# Splunk - exportera resultat

den 18 juni 2024 10:48

Att exportera ett resultat till en fil som excel kan läsa:




Öppna excel och välj data-> från text
Välj komma som avgränsare
=> Tabell som går att formattera som man vill

# Docker, Kubernetes, CI/CD system

den 1 juli 2024 20:40

**Docker med mina ord**
Med docker kan man paketera applikationer med alla dess beroenden, detta kan sedan deployas i en container.
För att paketera applikationen behövs en dockerfile, och så bygger man en image från den.

Man säger sedan till docker att starta en container med denna image.

**För att skapa en image och starta en container**
Skapa en dockerfile, dockerfilen har all info för att paketera applikationen som en image
Låt docker paketera applikationen:
**docker build -t hello-docker .**
Ger imagen en tag (-t), ett namn (hello-docker) och sparar imagen "här".

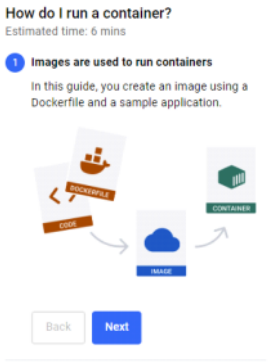
Alla hello-docker images:
**docker image -ls**

Säg sedan till docker att starta en container med denna image
**docker run hello-docker**

**Dvs**
- Man ***deployar*** en applikation ***i*** en container
- Man ***kör/startar*** en container med en viss image

**Fördelar**
- Mer lättviktigt än en virtual machine
- Har med sig allt som behövs, så det blir exakt samma förutsättningar / miljö varje gång imagen körs.
- Man kan starta flera containrar i samma miljö, genom docker-compose, då pekar man ut de images som ska köras. De blir då i samma nätverkscontext

**Vi hade docker för att läsa loggar**
- Ladda ner loggar fr servern till en katalog
- Köra splunk i docker
  - Första gången: Starta via gitbash (docker-compose up)
  - I docker desktop finns tabbar för att se containers resp images. Nu kan man starta och stoppa containern med splunk i därifrån.
- Vi kan enkelt rensa filerna i katalogen och splunk indexerar på nytt.




**Docker** används för att skapa och köra containers
**Kubernetes** används för att managera och automatisera deployment processen, skalning och hantering av containtar i kluster

Docker is a popular containerization platform that enables developers to package applications and their dependencies into lightweight, portable containers.

Kubernetes, often referred to as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. It focuses on managing clusters of containers, providing advanced features for workload distribution, scaling, and fault tolerance. Kubernetes abstracts away the underlying infrastructure, allowing developers to focus on application logic rather than the complexities of deployment and scaling.

When integrating Kubernetes with Docker, Kubernetes serves as the orchestration layer for containers created by Docker. Here's an overview of how they interact:

- **Container Creation**: Docker packages the application and its environment into a container, which is a lightweight, standalone, and executable software package.
- **Cluster Scheduling**: Kubernetes schedules these containers to run on various nodes in a cluster, taking into account the compute resources required by each container.
- **Scaling**: Kubernetes can automatically scale the number of containers up or down based on the workload demand, ensuring that the application can handle the incoming traffic and workloads efficiently.
- **Service Discovery and Load Balancing**: Kubernetes groups sets of containers and provides them with a single DNS name for service discovery and can load-balance the traffic among these containers.
- **Storage Orchestration**: Kubernetes automatically mounts the storage system of your choice, whether from local storage, a public cloud provider, or a network storage system, to containers.
- **Self-healing**: Kubernetes constantly monitors containers and replaces containers that fail, kill containers that don't respond to user-defined health checks, and doesn't advertise them to clients until they are ready to serve.
- **Automated Rollouts and Rollbacks**: Kubernetes helps you roll out changes to your application or its configuration while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes can rollback the change for you.

**CI/CD system**
Ett CI/CD system består av:
- Job server - där tex Jenkins körs
- Job scheduler - Jenkins är en job scheduler
- Job executor - Alla CI/CD system är en job executor engine

Ex: Tekton is a Kubernetes native CI/CD system, uses Kubernetes as much as it can, using its scheduler etc. => Behöver kunna kubernetes för att ens förstå termerna…

**Kubernetes = Job execution system**
Has a **scheduler**, that takes care of placing the pods on the nodes.
The pods can be considered as a job **executor**.
The node, where the pod is running, can be considered a job **server**

Kubernetes består av
- Control plane
- Nodes
- Pods

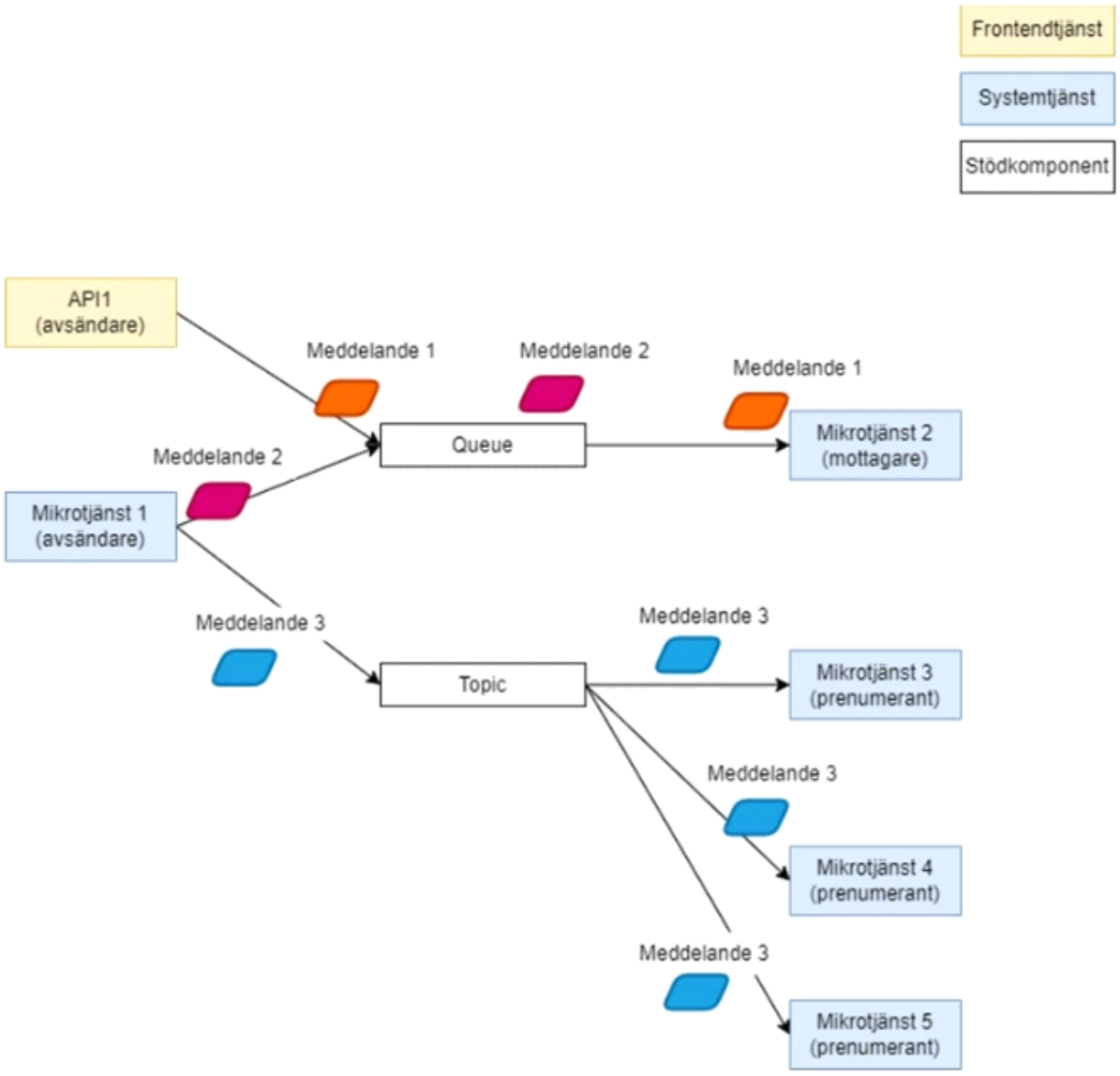
Pods: Organiserar containrar i pods.
Nodes: De maskiner som kör containrarna (Fysisk maskin eller vm)
Control plane: Managerar systemets övergång state (scheduling, scaling, konfiguration av kubernetes api server)

Läsa mer på denna sida för att lära sig mer om Tekton och Kubernetes
https://mkdev.me/posts/what-is-tekton-and-how-it-compares-to-jenkins-gitlab-ci-and-other-ci-cd-systems

# Kö

den 9 november 2023 14:55

Gör om detta till mina sidor



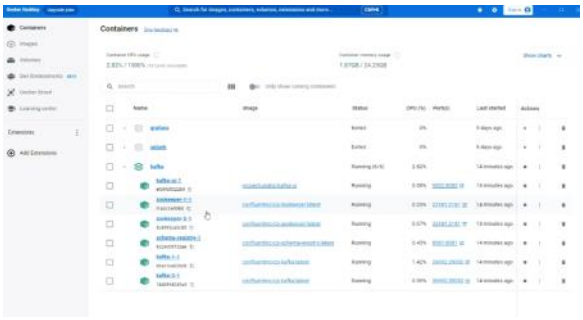

Exempelscenario meddelandehantering i mikrotjänstarkitektur

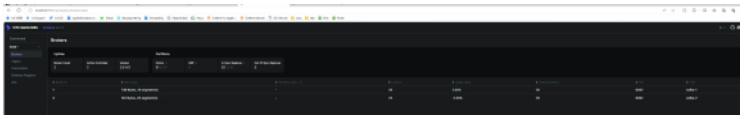
# Kafka

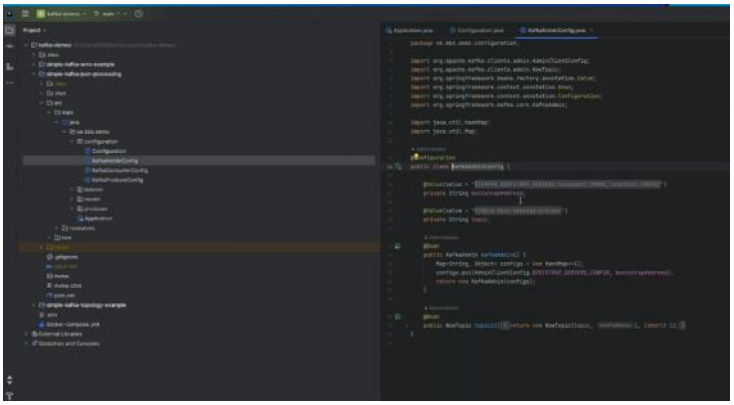
den 24 november 2023 09:37

SoftwareMill Kafka Visualization
Apache Kafka®: Basic Concepts, Architecture, and Examples (confluent.io) (grundkurs)
Kafka Streams: Basic Concepts, Architecture, and Examples (confluent.io)
Apache Kafka

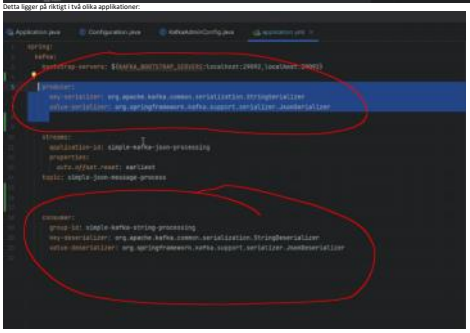
För varje topic (+kö):





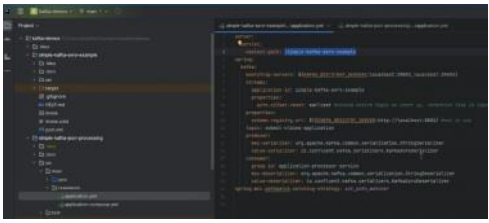


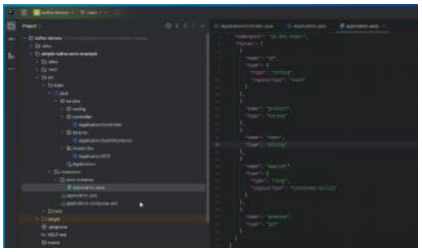
Detta ligger på riktigt i två olika applikationer:



Formatet i topicen får inte ändras!

Avro = json med historik/förändringar
*för liquibase*





Topologi = samverkan mellan köer

Två topics kombineras och så får man ett resultat.