Ostad

DOCKER

ইন্টারভিউ কোয়েশ্চেন

1 Hypervisor কী?

Hypervisor হলো এক ধরণ সফটওয়্যার যা Virtualization সম্ভব করে তোলে। একে Virtual Machine Monitor (VMM) ও বলা হয়।

2 Hypervisor কত ধরনের হয়?

Hypervisor দুই ধরনের হয়:

1. Type 1 Hypervisor:

- একে Native Hypervisor বা Bare Metal Hypervisorও বলা হয়।
- এটি সরাসরি Inharent Host System-এ চলে।
- এর Host System Hardware-এর উপর সরাসরি অ্যাক্সেস থাকে।
- Base Server Operating System-এর প্রয়োজন হয় না।
- উদাহরণ: Microsoft Hyper-V

2. Type 2 Hypervisor:

- একে Hosted Hypervisorও বলা হয়।
- এটি inherent Host Operating System ব্যবহার করে।
- Host Operating System-এর উপরে Application-এর মতো চলে।
- উদাহরণ:
- VMware Workstation
- VirtualBox

3 Virtualization কী?

Virtualization হলো একটি technology যা কম্পিউটিং resource, যেমন servers, storage devices, বা networks, এর মধ্যে ভার্চুয়াল version তৈরি করতে সক্ষম করে। এটি একটি ফিজিক্যাল মেশিনে একাধিক ভার্চুয়াল ইনস্ট্যান্স চালানোর সুযোগ দেয়, যা resource utilization এবং flexibility বাড়াতে সাহায্য করে।

Virtualization এর মাধ্যমে আপনি একটি System কে অনেকগুলো Partitions-এ ভাগ করতে পারেন যা Separate এবং Independent Systems-এর মতো কাজ করে। Hypervisor নামক একটি Software এই ধরধরণের Partitioning সম্ভব করে তোলে। Hypervisor দ্বারা তৈরি Virtual Inhabitants কে Virtual Machine বলা হয়।

4 Containerization কী?

Containerization হলো সফটওয়্যার অ্যাপ্লিকেশন এবং তাদের dependencies গুলি লাইটওয়েট, পোর্টেবল ইউনিট হিসেবে প্যাকেজ করা এবং ডিপ্লয় করা যেতে পারে। এই containers গুলি অ্যাপ্লিকেশনকে তার environment থেকে আলাদা করে তুলে ধরে, এটা বিভিন্ন কম্পিউটিং environment সহজে ডিপ্লয় করা করতে সাহায্য করে, যেমন ডেভেলপমেন্ট, টেস্টিং, এবং প্রোডাকশন।

5 Virtualization এবং Containerization এর মধ্যে পার্থক্য কী?

Virtualization:

- Hypervisor গেস্টকে একটি সম্পূর্ণ Virtual Machine (Kernel সহ) প্রদান করে।
- Virtual Machine-গুলো Physical Hardware layer একটি Abstraction।
- প্রতিটি VM একটি Machine।

Containerization:

- Container-এ application চালানোর জন্য একটি Isolated Environment প্রদান করে।
- সম্পূর্ণ User Space application -কে explicitly Dedicated করা হয়।
- Container-এর ভিতরে করা কোনো পরিবর্তন Host বা একই Host-এ রানিং অন্যান্য Containers-এ কখনোই reflected হয় না।
- Container হল Application Layer-এর একটি Abstraction।
- প্রতিটি Container একটি Application।

উদাহরণ: ধরা যাক, আপনার একটি Web Application আছে।

Virtualization:

আপনি Application-টি একটি Virtual Machine-এ Install করতে পারেন।

Virtual Machine-টি Host Machine-এর Hardware-এর উপরে চলবে।

Virtual Machine-টি Host Machine-এর Hardware-এর Isolated থাকবে।

Containerization:

আপনি Application-টি একটি Container-এ Package করতে পারেন।

Container-টি Host Machine-এর Isolated Environment-এ চলবে।

Container-টি Host Machine-এর Hardware-এর উপরে Directly চলবে।

6 Docker কী?

Docker একটি Open Source Containerization Platform। এটি Lightweight, Portable Containers ব্যবহার করে যেকোনো Application-এর Deployment সহজ করে তোলে।

Ostad

7 Docker এর সবচেয়ে উল্লেখযোগ্য কনফিগারেশন গুলি কী কী?

1. Application Agility:

- Docker Images-এর মাধ্যমে, Applications-কে সহজে Deploy এবং Update করা যায়।
- Applications-কে যেকোনো Environment-এ স্থানান্তর করা যায়।

2. Development Productivity:

- Docker Developers-কে Code-এর পরিবর্তে Configuration-এর উপর focus দিতে সাহায্য করে।
- Docker-এর মাধ্যমে, Developer-রা সহজে Local Environment-এ Applications Develop করতে পারে।

3. Simple modeling:

- Docker-এর মাধ্যমে, Applications-কে সহজে Model করা যায়।
- Docker Images-এর মাধ্যমে, Applications-এর Dependencies-কে Manage করা সহজ।

4. Operational efficiency:

- Docker Operations-কে Automate করতে সাহায্য করে।
- Docker-এর মাধ্যমে, Applications-কে সহজে Scale করা যায়।

5. Placement and Scaling:

- Docker Images-কে যেকোনো Infrastructure-এ Deploy করা যায়।
- Docker-এর মাধ্যমে, Applications-কে সহজে Scale Up এবং Scale Down করা যায়।

6. Version control:

- Docker Images-এর Version Control করা যায়।
- Docker-এর মাধ্যমে, Applications-এর Rollback করা সহজ।

৪ কেন কেউ Docker ব্যবহার করবে? এটা কী অফার করে?

Docker users গ্রহণ করে নেওয়ার জন্য অনেক inspiration দেয়, যেমন:

1. Quick and easy initial set up:

- Docker-এর মাধ্যমে, Applications-কে সহজে Install এবং Setup করা যায়।
- Docker-এর জন্য কোনো Special Hardware-এর প্রয়োজন নেই।

2. Application lifecycle বিস্তারিত বর্ণনা:

- Docker Images-এর মাধ্যমে, Applications-এর Dependencies-কে Manage করা সহজ।
- Docker Images-এর Version Control করা যায়।

3. Docker কম্পোজ সাথে সহজ configuration এবং Smooth interaction:

- Docker Compose-এর মাধ্যমে, Complex Applications-কে সহজে Configure করা যায়।
- Docker Compose-এর মাধ্যমে, Applications-কে সহজে Deploy করা যায়।

4. Complete and well-detailed documentation:

- Docker-এর well-detailed ডকুমেন্টেশন আছে।
- Docker-এর জন্য অনেক Community Support আছে।

5. পিসি বা এন্টারপ্রাইজ আইটি সিস্টেমে সমান স্বাচ্ছন্দ্যে চালানোর ক্ষমতা:

- Docker-কে যেকোনো Environment-এ Deploy করা যায়।
- Docker-কে On-Premises বা Cloud-এ Deploy করা যায়।

9 কেন Docker শিখবেন?

1. Development Process Complex এবং Challenging:

- Application development শুধু কোড লেখার চেয়ে অনেক বেশি।
- এতে বিভিন্ন ফ্রেমওয়ার্ক এবং আর্কিটেকচার ব্যবহার করা হয়।
- এটি ডেভেলপার জন্য process টিকে আরও complex করে তোলে।

2. Containerization development কে সহজ করে:

- Developer তাদের পছন্দ technology ব্যবহার করার স্বাধীনতা দেয়।
- Application ওয়ার্কফ্লোকে সহজ করে তোলে।

3. DevOps-এর জন্য Containerization গুরুত্বপূর্ণ:

- Developer প্রোডাক্টিভিটি বৃদ্ধি করে।
- Application এর স্কেলেবিলিটি বৃদ্ধি করে।
- Resources পরিচালনাকে আরও দক্ষ করে তোলে।

4. Docker হলো একটি জনপ্রিয় Containerization প্ল্যাটফর্ম:

- এটি Light-Weight এবং ব্যবহার করা সহজ।
- এটি Application ডেভেলপমেন্ট এবং ডেপ্লয়মেন্টকে সহজ করে তোলে।
- PayPal, Spotify, Uber-এর মতো অনেক কোম্পানি Docker ব্যবহার করে।

10

আপনি Docker Container সম্পর্কে কিছু বলতে পারেন?

- Docker Container-এ application এবং তার সমস্ত dependencies থাকে।
- Container-গুলো Kernel এবং System Resources ভাগ করে এবং Host Operating System-এ Isolated System হিসাবে চালায়।
- যেকোনো Platform-এ Containerized Application Run করা।
- Docker Container-গুলো Docker Image-এর Runtime Instance

11 Docker এর কোন downside আছে?

Docker-এর downside:

1. স্টোরেজ অপশন অভাব:

- Docker-এর নিজস্ব স্টোরেজ সিস্টেম নেই।
- Docker-এর জন্য External Storage Solution ব্যবহার করতে হবে।

2. মনিটরিং অপশন অভাব:

- Docker-এর নিজস্ব মনিটরিং সিস্টেম নেই।
- Docker-এর জন্য Third-Party Monitoring Tools ব্যবহার করতে হবে।

3. ইনএক্টিভ নোডগুলি Automatic ভাবে পুনরায় নির্ধারণ করার inability:

- Docker-এর মাধ্যমে ইনএক্টিভ নোডগুলি Automatic ভাবে পুনরায় নির্ধারণ করা সম্ভব নয়।
- ইনএক্টিভ নোডগুলি Manually Restart করতে হবে।

4. Automatic Horizontal স্কেলিং সেট আপ জটিলতা:

- Docker-এর মাধ্যমে Automatic Horizontal স্কেলিং সেট আপ করা জটিল।
- Automatic Horizontal স্কেলিং সেট আপ করার জন্য Third-Party Tools ব্যবহার করতে হবে।

12 বিভিন্ন Docker component ব্যাখ্যা করুন।

তিনটি প্রধান Docker component হল:

1. Docker Client:

- Docker Host-এর সাথে যোগাযোগ করে।
- Docker Build, Run, Pull, Push ইত্যাদি অপারেশন সম্পাদন করে।
- Docker API ব্যবহার করে Docker Daemon-এর সাথে কথা বলে।

2. Docker Host:

- Docker Daemon চালায়।
- Containers এবং Images-কে Manage করে।
- Registry-এর সাথে সংযোগ করে।

3. Registry:

- Docker Images-কে Store করে।
- Public এবং Private Registry থাকে।
- Docker Hub এবং Docker Cloud হল দুটি জনপ্রিয় Public Registry।

13 Docker image কী?

Docker image হল Docker Container উৎস। অন্য কথায়, Docker image গুলি Container তৈরি করতে ব্যবহার করা হয়। যখন একজন ব্যবহারকারী একটি Docker image চালায়, তখন একটি Container instance তৈরি হয়। এই Docker image যে কোনো Docker ইনভায়রনমেন্ট-এ deploy করা যেতে পারে।

14 Docker ফাইল কী?

এটি একটি টেক্সট ফাইল যাতে সমস্ত কমান্ড রয়েছে যা একটি প্রভাইডেড docker image তৈরির জন্য লিখতে হবে।

15 Docker Hub কী?

Docker image , Docker Container তৈরি করে। এই Docker image গুলি যেখানে থাকে সেখানে একটি রেজিস্ট্রি থাকতে হবে। এই রেজিস্ট্রি হল Docker Hub. ব্যবহারকারীরা Docker Hub থেকে image তুলতে পারে এবং কাস্টমাইজড image এবং Container তৈরি করতে ব্যবহার করতে পারে। বর্তমানে, Docker Hub হল docker image container গুলির বিশ্ব বৃহত্তম পাবলিক ভান্ডার।

16 Docker আর্কিটেকচার ব্যাখ্যা কর।

Ostad

Docker আর্কিটেকচার তিনটি প্রধান component নিয়ে গঠিত:

1. Docker Daemon:

- এটি একটি Long-Running Program যা Server হিসেবে কাজ করে।
- Docker Commands Daemon কে কী করতে হবে তা নির্দেশ করে।

2. REST API:

- এটি Programs Daemon এর সাথে কথা বলার জন্য ব্যবহার করে।
- CLI এবং অন্যান্য Docker Applications API ব্যবহার করে Daemon এর সাথে Interact করে।

3. CLI (Command Line Interface):

- এটি Users-কে Docker Commands দিয়ে Docker Engine-এর সাথে Interact করার মাধ্যম প্রদান করে।
- CLI Scripting-এর জন্যও ব্যবহার করা যেতে পারে।

17 Docker শেখার সুবিধা কী?

Docker শেখার সুবিধা:

- Portability: Docker Container গুলিকে যেকোনো প্ল্যাটফর্মে deploy করা যায়।
- Scalability: Docker Container গুলিকে সহজে স্কেল করা যায়।
- Performance: Docker রিসোর্স ব্যবহারকে Optimize করে।
- Security: Docker application security বৃদ্ধি করে।

18 Docker Compose সম্পর্কে কিছু বলুন।

Docker Compose, একটি YAML ফাইল যাতে Docker application সেট আপ করার জন্য সার্ভিস, নেটওয়ার্ক এবং ভলিউম সম্পর্কে ডিটেলস রয়েছে। সুতরাং, আপনি আলাদা Container তৈরি করতে, হোস্ট করতে এবং একে অপর সাথে যোগাযোগ করতে Docker কম্পোজ ব্যবহার করতে পারেন। প্রতিটি Container অন্য Container সাথে যোগাযোগ জন্য একটি পোর্ট এক্সপোজ করবে।

19 Docker Swarm কী?

Docker Swarm হল Docker-এর জন্য Native Clustering Solution। এটি Docker Hosts-এর একটি Pool-কে সিঙ্গেল , Virtual Docker Host-এ পরিণত করে।

20 Docker নেমস্পেস কী?

নেমস্পেস হল লিনাক্স একটি কনফিগারেশন যা Processes-কে Isolate করার জন্য ব্যবহার করা হয়। Container-এর ক্ষেত্রে, নেমস্পেস Isolation-এর একটি Layer যুক্ত করে।

Docker বিভিন্ন ধরণ নেমস্পেস সরবরাহ করে যা Containers-কে Portable করে তোলে এবং Host System-কে Affect না করে।

21 একটি Docker Container Life-cycle বর্ণনা করুন।

একটি Docker Container Life-cycle:

1. Container Create:

- Docker Image-এর মাধ্যমে Container তৈরি করা হয়।
- Docker Image হলো Read-only Template।

2. Container Run:

- docker run Command-এর মাধ্যমে Container চালানো হয়।
- Container-এর Startup Process শুরু হয়।

3. Container Break(অপশনাল):

- docker pause Command-এর মাধ্যমে Container Pause করা হয়।
- Container Paused State-এ চলে যায়।

4. Unpause Container:

- docker unpause Command-এর মাধ্যমে Container Unpause করা হয়।
- Container Running State-এ ফিরে আসে।

5. Container Start :

- docker start Command-এর মাধ্যমে Container Start করা হয়।
- Container Running State-এ চলে যায়।

6. ContainerStop:

- docker stop Command-এর মাধ্যমে Container Stop করা হয়।
- Container Exited State-এ চলে যায়।

7. Container Resart:

- docker restart Command-এর মাধ্যমে Container Restart করা হয়।
- Container Stop করা হয় এবং তারপর Start করা হয়।

8. Container kill:

- docker kill Command-এর মাধ্যমে Container Kill করা হয়।
- Container Forcefully Terminated হয়।

9. Container destroy:

- docker rm Command-এর মাধ্যমে Container Destroy করা হয়।
- Container Deleted হয়।

22 Docker মেশিন কি?

Docker মেশিন একটি tool যা আপনাকে ভার্চুয়াল হোস্টে Docker ইঞ্জিন ইনস্টল করতে দেয়। এই হোস্টগুলি এখন Docker -মেশিন কমান্ড ব্যবহার করে পরিচালনা করা যেতে পারে। Docker মেশিন আপনাকে Docker Swarm ক্লাস্টার চালাতে করতে দেয়।

23

প্রয়োজনীয় Docker কমান্ড নাম এবং তারা কি করে।

সবচেয়ে গুরুত্বপূর্ণ Docker কমান্ড হল:

- Build: একটি Docker Image ফাইল তৈরি করে
- Commit: Container পরিবর্তন থেকে একটি নতুন Image তৈরি করে
- Create: একটি নতুন Container তৈরি করে
- Dockerd: Docker Daemon চালু করে
- Kill: একটি Container kill করে

24 Docker অবজেক্ট লেবেল কি?

লেবেল হল Container, image, Local ডেমন, নেটওয়ার্ক, ভলিউম এবং নোড মতো Docker অবজেক্টে মেটাডেটা প্রয়োগ করার পদ্ধতি।

25 Docker কিভাবে VMware বা VirtualBox এর মত প্রথাগত ভার্চুয়ালাইজেশন technology থেকে আলাদা?

VMware এবং VirtualBox:

- Hypervisor ব্যবহার করে।
- Complete Virtual Machines চালায়।
- Resource Intensive।
- Slow Startup Time।

Docker:

- Containerization ব্যবহার করে।
- Applications-কে Isolated Containers-এ চালায়।
- Resource Efficient।
- Fast Startup Time।

কারণ:

- VMware এবং VirtualBox প্রতিটি Virtual Machine-এর জন্য Full Operating System Install করে।
- Docker Host Operating System-কে Share করে।

সুবিধা:

- Docker Lightweight এবং Portable।
- Docker Applications-কে Deploy এবং Scale করা সহজ।

26 Docker image এবং Docker Container মধ্যে পার্থক্য কি?

Docker image :

- একটি লাইটওয়েট, এক্সিকিউটেবল সফ্টওয়্যার প্যাকেজ।
- Application চালানোর জন্য প্রয়োজনীয় সমস্ত কিছু ইনক্লুড করে।
- Read-only এবং Immutable।

Docker Container:

- Docker image রানটাইম Instance।
- Isolated Environment-এ চলে।
- Read-write এবং Mutable।

সহজভাবে বলতে গেলে:

- Docker image হল Blueprint।
- Docker Container হল Building।

26 Docker বিল্ড process বর্ণনা করুন।

Docker বিল্ড process দুটি মূল ধাপ জড়িত:

1. Dockerfile লেখা:

- Dockerfile হলো একটি Text Document যা Docker Image তৈরির জন্য প্রয়োজনীয় Instructions ধারণ করে।
- Dockerfile-কে Image তৈরির Blueprint বলা হয়।
- Dockerfile-এ Instructions লেয়ার-ভিত্তিক।

2. Docker Build Command চালান:

- Docker Build Command Dockerfile-কে Input হিসেবে নেয়।
- Docker Build Command Dockerfile-এর Instructions অনুসারে Image তৈরি করে।
- Image-এর প্রতিটি Layer Dockerfile-এর একটি Instruction থেকে তৈরি হয়।
- Layers-কে Stack করে Final Image তৈরি করা হয়।

27

আপনি কিভাবে সাইজ এবং পারফরমেন্স জন্য Docker image অপ্টিমাইজ করবেন?

Docker image অপ্টিমাইজ করার কৌশল:

1. মাল্টি-স্টেজ বিল্ড ব্যবহার করুন:

- Application তৈরির জন্য একটি Image এবং Running-এর জন্য আলাদা, Lightweight Image ব্যবহার করুন।
- চূড়ান্ত Image Build Tools এবং Intermediate Artifacts বহন করে না, যার ফলে Size কমে।

2. লেয়ারগুলি ছোট করুন:

- প্রতিটি RUN, COPY, বা ADD Command Image-এ একটি নতুন Layer তৈরি করে।
- Layer-এর সংখ্যা কমাতে যেখানে সম্ভব এই Commands-গুলোকে একত্রিত করে।

3. Dockerignore ফাইল ব্যবহার করুন:

- Image তৈরির সময় ignore করার জন্য File বা Directory নির্দিষ্ট করুন।
- অপ্রয়োজনীয় বা Sensitive Data ইনক্লুড করা থেকে বাধা দেয়।

4. একই লেয়ারে clean করুন:

- Dockerfile Command-এর মধ্যে Package Manager দ্বারা তৈরি অপ্রয়োজনীয় File-গুলি সরান।

5. ছোট Base image ব্যবহার করুন:

- Base Image হিসাবে সম্পূর্ণ OS ব্যবহার করার পরিবর্তে, ছোট, Focused Base Image ব্যবহার করুন।
- Image Size উল্লেখযোগ্যভাবে কমে।

উপরোক্ত কৌশলগুলি ব্যবহার করে, আপনি আপনার Docker Images-এর Size এবং Performance বৃদ্ধি করতে পারেন।

29

Docker নেটওয়ার্কিং এর ধারণা ব্যাখ্যা করুন এবং Docker availability বিভিন্ন নেটওয়ার্ক ড্রাইভার নিয়ে আলোচনা করুন।

Docker নেটওয়ার্কিং:

Docker নেটওয়ার্কিং Container গুলিকে একে অপর সাথে এবং অন্যান্য সিস্টেম সাথে যোগাযোগ করতে দেয়।

Docker বিভিন্ন নেটওয়ার্ক ড্রাইভার availability:

1. Bridge:

- এটি ডিফল্ট নেটওয়ার্ক ড্রাইভার।
- এটি হোস্টে একটি ব্যক্তিগত internal নেটওয়ার্ক সরবরাহ করে।
- Container গুলি সহজেই একে অপর সাথে যোগাযোগ করতে পারে।

2. Host:

- Container নেটওয়ার্ক স্ট্যাকটি Docker হোস্ট থেকে বিচ্ছিন্ন হয় না।
- Container টি সরাসরি Docker হোস্ট নেটওয়ার্কিং ব্যবহার করে।
- এটি Bridge ড্রাইভার চেয়ে দ্রুত।
- Container টি তার পোর্টগুলিকে সরাসরি বাইর বিশ্ব কাছে প্রকাশ করে।

3. Overlay:

- Docker Swarm-এ ব্যবহার করা হয়।
- Swarm service গুলি একে অপর সাথে যোগাযোগ করতে পারে।
- একাধিক হোস্ট জুড়ে বণ্টন করা নেটওয়ার্ক এর জন্য আদর্শ।

4. MacVLAN:

- Container টিকে নেটওয়ার্কে ফিজিক্যাল ডিভাইস হিসাবে deploy করে।
- Docker ডেমন কনটেইনার এবং External নেটওয়ার্কগুলির মধ্যে ট্র্যাফিককে রাউট করে।

5. None:

- Container জন্য সমস্ত নেটওয়ার্কিং বাতিল করে।

30

Docker ভলিউম কি এবং কিভাবে এটি Container জুড়ে ডেটা consistency রক্ষা করে?

Docker ভলিউম:

- Docker Container গুলির দ্বারা জেনারেটেড এবং ব্যবহৃত ডেটা স্থায়ী করার জন্য পছন্দ process।
- ডিফল্টরূপে, Docker Container গুলি স্টেটলেস।
- ডেটা Container বাইরে হোস্ট সিস্টেমে সংরক্ষণ করা হয়।
- ডেটা Container Life-cycle থেকে স্বাধীন।
- ডেটা নিরাপদে একাধিক Container কে ভাগ করা যেতে পারে।

Docker ভলিউম কিভাবে Container জুড়ে ডেটা consistency সক্ষম করে:

- ডেটা Container থেকে আলাদা করে।
- ডেটা Container মুছে ফেলা থেকে রক্ষা করে।
- ডেটা Container জীবনকাল জুড়ে স্থায়ী করে।
- ডেটা একাধিক Container রে ভাগ করা সহজ করে।

31 Docker ফাইলস এবং ক্যাশিং মেকানিজম ভূমিকা কি?

Docker ফাইল:

- Docker Image তৈরি করার জন্য নির্দেশাবলী ধারণ করে।
- Base Image, Commands, Ports, Environment Variables ইত্যাদি নির্ধারণ করে।

Docker ফাইল গুরুত্বপূর্ণ নির্দেশাবলী:

- FROM: Base Image নির্ধারণ করে।
- LABEL: Image-এর Metadata নির্ধারণ করে।
- RUN: Commands Run করে।
- ADD: Files or Directories Copy করে।
- EXPOSE: Ports Open করে।
- CMD/ENTRYPOINT: Container Startup-এ Command Run করে।

Docker ক্যাশিং:

- Docker Image Build Process-কে Speed Up করে।
- Dockerfile-এর প্রতিটি Line-এর Result Cache করে।
- File Unchanged থাকলে, Cache-এর Layer-গুলো Reuse করে।

32

Docker এ আমার কম্পোজ ফাইল জন্য আমি কি YAML এর পরিবর্তে JSON ব্যবহার করতে পারি?

হ্যাঁ, আপনি আপনার Docker কম্পোজ ফাইল জন্য ডিফল্ট YAML এর পরিবর্তে JSON ব্যবহার করতে পারেন। কম্পোজিংয় সাথে JSON ফাইল ব্যবহার করার জন্য, আপনাকে নিম্নলিখিত হিসাবে ব্যবহার করার জন্য ফাইল নাম নির্দিষ্ট করতে হবে:

```
docker-compose -f docker-compose.json up
```

33 আপনি কিভাবে Docker Container সিকিউর করতে পারেন এবং পোটেনশিয়াল উইকনেসে থেকে কিভাবে রক্ষা করতে পারেন?

Docker Container গুলির নিরাপত্তা নিশ্চিত করতে একাধিক সর্বোত্তম অনুশীলন প্রয়োজন।

- উইকনেসে কমাতে বিশ্বস্ত এবং ন্যূনতম বেস image ব্যবহার করুন।
- সর্বদা Docker অফিসিয়াল রিপোজিটরির মতো বিশ্বস্ত সোর্স থেকে Docker image গুলি টেনে আনুন।
- সম্ভব হলে রাউট হিসাবে Container বা সার্ভিসেগুলি চালানো এড়িয়ে চলুন।
- Docker envirtonmentআপ টু ডেট রাখুন।
- Container এবং external নেটওয়ার্ক মধ্যে নিরাপদ যোগাযোগ। প্রয়োজনে এনক্রিপশন জন্য TLS ব্যবহার করুন।
- প্রাথমিক উইকনেস সনাক্তকরণ এবং দ্রুত প্রতিক্রিয়ার জন্য সিকিউরিটি স্ক্যানিং এবং Review tool গুলি প্রয়োগ করুন।
- নিয়মিত নিরাপত্তা অডিট এবং রিভিউ ম্যানেজমেন্ট করুন।

34 কিভাবে আপনি পূর্বে Docker ব্যবহার করেছেন?

এটি এমন একটি প্রশ্ন যা আপনি Docker সাথে আপনার সম্পূর্ণ অভিজ্ঞতা নিয়ে আসতে পারেন এবং আপনি যদি Docker আগে অন্য কোন কনটেইনার technologyব্যবহার করে থাকেন। আপনি আরও ব্যাখ্যা করতে পারেন যে এই technology টি ম্যানুফ্যাকচারিং Life-cycle ব্যবdeployment র বিকাশ Automatic তা এনেছে। আপনি Docker সাথে যেমন পাপেট, শেফ বা এমনকি সমস্ত technology র মধ্যে সবচেয়ে জনপ্রিয় - জেনকিন্স সাথে কাজ করেছেন এমন অন্য কোনও ইন্টিগ্রেশন নিয়েও আপনি আলোচনা করতে পারেন। যদি আপনার কাছে Docker নিজে থেকে কিন্তু এই স্থান থেকে অনুরূপ tool গুলির সাথে কোনও অভিজ্ঞতা না থাকে তবে আপনি একই কথা জানাতে পারেন এবং এই লিডিং Container technology শেখার জন্য আপনার আগ্রহও দেখাতে পারেন।

35 Container Orchestration পরিচালনায় Docker Swarm এবং Kubernetes ভূমিকা বর্ণনা করুন।

Container Orchestration পরিচালনায় Docker Swarm এবং Kubernetes ভূমিকা:

Docker Swarm:

- Docker-এর Local Orchestration tool
- Docker ইঞ্জিন সাথে নির্বিচ্ছিন্নভাবে কাজ করে
- সহজ সেটআপ এবং ব্যবহার
- ছোটো ও মাঝারি আকার application জন্য আদর্শ

Kubernetes:

- ক্লাউড-নেটিভ Orchestration প্ল্যাটফর্ম
- Kubernetes large-scale, জটিল অ্যাপ্লিকেশনের জন্য একটি জনপ্রিয় পছন্দ।
- কুবারনেটস Docker Swarm চেয়ে আরও বিস্তৃত features এবং কাস্টমাইজেশন বিকল্প সরবরাহ করে।

36 Docker registry এবং Docker Repository এর মধ্যে পার্থক্য ব্যাখ্যা করুন।

Docker registry এবং Docker repository-এর মধ্যে পার্থক্য:

Registry:

- Images store করে।
- Public অথবা private হতে পারে।
- Docker hub হল public registry-এর উদাহরণ।

Repository:

- একই নাম images-এর collection।
- Tag দ্বারা আলাদা করা হয়।
- "debian:10" হল repository-তে image-এর উদাহরণ।

37

একটি Docker Container কে একটি ভিন্ন ভার্সনে আপগ্রেড বা রোলব্যাক করার ক্ষেত্রে জড়িত পদক্ষেপগুলি বর্ণনা করুন।

Docker Container আপগ্রেড বা রোলব্যাক করার পদক্ষেপ:

১. image নতুন ভার্সন টানুন:

docker pull <image-name>:<new-version>

২. বন্ধ করুন এবং বিদ্যমান Container টি সরান:

docker stop <Container-name>

docker rm <Container-name>

৩. নতুন Container চালান:

docker run -d <image-name>:<new-version>

38

Docker মাল্টি-স্টেজ বিল্ড ব্যবহার করার সুবিধা এবং ব্যবহার ক্ষেত্রে কী কী?

মাল্টি-স্টেজ বিল্ড Docker একটি শক্তিশালী টেকনিক যা আপনাকে সিকিউর, স্কীলড এবং অপ্টিমাইজড image তৈরি করতে সাহায্য করে। এটি একাধিক Docker ফাইল ব্যবহার করে, প্রতিটি একটি নতুন স্টেজ সূচনা করে।

সুবিধা:

- image সাইজ হ্রাস: মধ্যবর্তী স্টেজগুলি আর্টওয়ার্ক তৈরি করতে পারে যা পরবর্তী স্টেজে কপি করা হয়,
- সিকিউরিটি বৃদ্ধি: চূড়ান্ত image টি ছোট, এটাক সারফেস হ্রাস করে,
- ডিপেন্ডেন্সি ডিপ্লয়মেন্ট : বিভিন্ন স্টেজ বিভিন্ন ডিপেন্ডেন্সি ব্যবহার করতে পারে,
- বিল্ড process দ্রুততর: Docker ক্যাশে করা লেয়ার ব্যবহার করতে পারে।

ব্যবহার ক্ষেত্র:

- image সাইজ মিনিমাইজ করা
- সিকিউরিটি ডেভেলপ করা
- বিল্ড-টাইম এবং রানটাইম ডিপেন্ডেন্সি আলাদা করা
- বিল্ড আপ এক্সিলারেটেড করা

39 Docker --privileged Flag ব্যবহার প্রভাব এবং কেন এটি সতর্কতার সাথে ব্যবহার করা উচিত তা আলোচনা করুন।

Docker --privileged Flag ব্যবহার প্রভাব এবং সতর্কতার সাথে ব্যবহার প্রয়োজনীয়তা:

--privileged flag:

- এটি একটি শক্তিশালী বিকল্প যা Container কে প্রায় হোস্ট মেশিন মতো একই সুবিধা দেয়।
- Container টি হোস্ট সিস্টেম রিসোর্স, ডিভাইস ড্রাইভার এবং কার্নেল মডিউলগুলি অ্যাক্সেস এবং ম্যানিপুলেট করতে পারে।

সতর্কতার সাথে ব্যবহার কারণ:

- হোস্টকে নিরাপত্তা ঝুঁকিতে প্রকাশ করতে পারে।
- Docker containaization মডেল দ্বারা প্রদত্ত নিরাপত্তা ব্যবস্থাগুলিকে ওভাররাইড করে।
- মূলত হোস্ট সিস্টেম থেকে application process এবং এনভায়রনমেন্ট আইসোলেত করার উদ্দেশ্যকে হারায়।

40

কিভাবে Docker resource লিমিটেশন এবং আইসোলেশন যেমন CPU, মেমরি এবং I/O পরিচালনা করে?

Docker বিভিন্ন process ব্যবহার করে Container resource লিমিটেশন এবং আইসোলেশন করে, যেমন:

CPU:

- নির্দিষ্ট সংখ্যক CPU শেয়ার বরাদ্দ করা।
- নির্দিষ্ট CPU বা কোর ব্যবহার নির্ধারণ করা।
- --cpu-shares, --cpuset-cpus, --cpu-period ব্যবহার করা।

মেমরি:

- --memory বিকল্প ব্যবহার করে মেমরি সীমাবদ্ধ করা।
- মেমরি সংরক্ষণ সীমা নির্ধারণ করতে --memory-reservation ব্যবহার করা।

I/O:

- হোস্ট OS-এর I/O সময়সূচী ব্যবহার করা।
- --blkio-weight, --device-read-bps, --device-write-bps ব্যবহার করে I/O ওজন এবং সীমা নির্ধারণ করা।

41

Docker ক্লায়েন্ট এবং Docker সার্ভার ভার্সন কিভাবে টেস্ট করবেন?

নিম্নলিখিত কমান্ড আপনাকে Docker ক্লায়েন্ট এবং সার্ভার cerison সম্পর্কে তথ্য দেয়:

$ docker version

42 আপনি কীভাবে রানিং, বিরতি দেওয়া এবং থামানো Container গুলির সংখ্যা পাবেন?

আপনার সিস্টেমে ইনস্টল করা Docker সম্পর্কে বিস্তারিত তথ্য পেতে আপনি নিম্নলিখিত কমান্ডটি ব্যবহার করতে পারেন।

$ docker info

43

আপনি যদি অস্পষ্টভাবে কমান্ডটি মনে রাখেন এবং আপনি এটি নিশ্চিত করতে চান, তাহলে আপনি সেই নির্দিষ্ট কমান্ডে কীভাবে সহায়তা পাবেন?

নিম্নলিখিত কমান্ডটি খুব দরকারী কারণ এটি আপনাকে একটি কমান্ড, সিনট্যাক্স ইত্যাদি ব্যবহার করতে সহায়তা করে।

$ docker --help

উপর কমান্ডটি সমস্ত Docker কমান্ড লিস্ট করে। আপনার যদি একটি নির্দিষ্ট কমান্ড সাহায্য প্রয়োজন হয়, আপনি নিম্নলিখিত সিনট্যাক্স ব্যবহার করতে পারেন:

$ docker <command> --help

44 Docker রিপোজিটরিতে কিভাবে লগইন করবেন?

আপনি hub.docker.com এ লগইন করতে নিম্নলিখিত কমান্ডটি ব্যবহার করতে পারেন:

$ docker login

আপনাকে আপনার ব্যবহারকারীর নাম এবং পাসওয়ার্ড জন্য Request করা হবে, সেগুলি প্রবেশ করান এবং অভিনন্দন, আপনি লগ ইন করেছেন।

Ostad

45 আপনি যদি একটি বেস ইমেজ ব্যবহার করতে চান এবং পরিবর্তন করতে চান, তাহলে আপনি কি করবেন?

আপনি আপনার Local সিস্টেমে Docker হাব থেকে একটি image টানুন

Docker হাব থেকে একটি image টানতে এটি একটি সাধারণ কমান্ড:

$ docker pull <image_name>

46

আপনি কিভাবে একটি image থেকে একটি Docker Container তৈরি করবেন?

একটি ইমেজ থেকে একটি ডকার কন্টেইনার তৈরি করতে, আপনি ব্যবহার করুন 'docker run' কমান্ড দিয়ে তারপর image_name দিয়ে:

$ docker run image_name

এই কমান্ডটি image-টি যদি লোকালভাবে না থাকে তবে এটি তা নিয়ে একটি Image extraction করে এবং সেই image উপর ভিত্তি করে একটি কন্টেইনার চালু করে।

47 আপনি কিভাবে সমস্ত রানিং Container লিস্ট করবেন?

নিম্নলিখিত কমান্ড সমস্ত রানিং Container করে:

$ docker ps

48 ধরুন আপনার কাছে ৩টি Container চলছে এবং এর মধ্যে থেকে আপনি একটিতে প্রবেশ করতে চান। আপনি কিভাবে একটি রানিং Container অ্যাক্সেস করবেন?

নিম্নলিখিত কমান্ড একটি রানিং Container অ্যাক্সেস করতে দেয়:

$ docker exec -it <Container id> sh

exec কমান্ড আপনাকে একটি Container প্রবেশ করতে এবং এটির সাথে কাজ করতে দেয়।

49 কিভাবে একটি Container শুরু?

একটি Docker Container শুরু করতে নিম্নলিখিত কমান্ড ব্যবহার করা হয়:

$ docker start <container_id>

Ostad

50

আপনি একটি Container ব্যবহার করতে পারেন, এটি এডিট করতে পারেন এবং এটি আপডেট করতে পারেন? এছাড়াও, আপনি কীভাবে এটিকে একটি নতুন করে Local সিস্টেমে সংরক্ষণ করবেন?

অবশ্যই, আপনি একটি Container ব্যবহার করতে পারেন, এটি এডিট করতে পারেন এবং এটি আপডেট করতে পারেন।

$ docker commit <conatainer id> <username/imagename>

Ostad

51

একবার আপনি একটি image নিয়ে কাজ করলে, কিভাবে আপনি এটিকে Docker হাবে push করে দিবেন?

$ docker push <username/image name>

52 কিভাবে একটি বন্ধ Container মুছে ফেলা যায়?

একটি বন্ধ Container মুছে ফেলার জন্য নিম্নলিখিত কমান্ড ব্যবহার করুন:

$ docker rm <Container id>

53 কিভাবে একটি Container stop করবেন?

একটি রানিং Container বন্ধ করার জন্য নিম্নলিখিত:

$ docker stop <Container_id>

54 কিভাবে একটি Container kill করবেন?

নিম্নলিখিত কমান্ড দিয়ে একটি Container kill করবো:

$ docker kill <Container_id>

55

কিভাবে Local স্টোরেজ সিস্টেম থেকে একটি image মুছে ফেলা যায়?

নিম্নলিখিত কমান্ড আপনাকে Local সিস্টেম থেকে একটি image মুছতে দেয়:

$ docker rmi <image-id>

56 কিভাবে একটি Docker ফাইল তৈরি করবেন?

একবার আপনি একটি Docker ফাইল লিখলে, আপনাকে সেই কনফিগারেশন গুলির সাথে একটি image তৈরি করার সাথে এটি তৈরি করতে হবে। একটি Docker ফাইল তৈরি করতে নিম্নলিখিত কমান্ডটি ব্যবহার করুন:

$ docker build <path to docker file>

57

আপনি কখন ".dockerfile_name" ব্যবহার করবেন এবং কখন পুরো পথটি ব্যবহার করবেন?

Docker ফাইল একই ফাইল ডিরেক্টরি থেকে exit করার সময় ".dockerfile_name" ব্যবহার করুন এবং যদি এটি অন্য কোথাও থাকে তবে আপনি পুরো pathটি ব্যবহার করেন।

58 আপনি কি জানেন কেন Docker সিস্টেম prune ব্যবহার করা হয়? এটার কাজ কি?

$ docker system prune

উপর কমান্ডটি সমস্ত বন্ধ Container , সমস্ত নেটওয়ার্ক যা ব্যবহার করা হয় না, সমস্ত hanging image এবং সমস্ত বিল্ড ক্যাশে রিমুভ করতে ব্যবহৃত হয়। এটি সবচেয়ে দরকারী Docker কমান্ডগুলির মধ্যে একটি।

59 আপনি কীভাবে Docker ক্লায়েন্ট এবং সার্ভার ভার্সন গুলি চেক করবেন?

এই কমান্ডটি আপনাকে আপনার প্রয়োজনীয় সমস্ত তথ্য দেয়:

```
$ Docker version
```

60

বর্তমানে রানিং / Stopped সমস্ত Container আপনি কিভাবে লিস্ট করবেন?

কমান্ডটি ব্যবহার করুন:

```
$ docker ps -a
```

61 একটি Docker Container স্কেলিং করার সাথে কী জড়িত?

এটা সাধারণত Kubernetes বা Docker Swarm এর মতো orchestration টুল ব্যবহার করে করা হয়, যা অটোমেটিকভাবে containerized applications এর ডিপ্লয়মেন্ট, স্কেলিং, এবং মনিটরিং পরিচালনা করে। এই টুলগুলি নির্ধারিত নিয়ম বা মেট্রিক্সের উপর ভিত্তি করে সহজে যেমন প্রয়োজনে অতিরিক্ত containers তৈরি করা বা নেওয়া হয়, তেমনি সঠিক কার্যক্রম মেনে চলে।

62 Docker সিস্টেম redundancy সম্পর্কে আপনি কী জানেন?

এটি একটি কমান্ড যা সমস্ত বন্ধ Container , অব্যবহৃত নেটওয়ার্ক, ক্যাশে তৈরি এবং hanging images সরাতে ব্যবহৃত হয়। Docker সবচেয়ে দরকারী কমান্ডগুলির মধ্যে একটি হল redundancy সিনট্যাক্স হল:

$ Docker system redundency

63 কিভাবে Docker Container বন্ধ করে পুনরায় চালু করবেন?

নিম্নলিখিত কমান্ডটি Container আইডি সহ একটি নির্দিষ্ট Docker Container বন্ধ করতে ব্যবহার করা যেতে পারে

CONTAINER_ID:

docker stop CONTAINER_ID

নিম্নলিখিত কমান্ডটি Container আইডি সহ একটি নির্দিষ্ট Docker Container পুনরায় চালু করতে ব্যবহার করা যেতে পারে

CONTAINER_ID:

docker restart CONTAINER_ID

64 Docker কোন প্ল্যাটফর্মে চলে?

Docker বর্তমানে নিম্নলিখিত প্ল্যাটফর্মগুলিতে এবং নিম্নলিখিত বিক্রেতা বা লিনাক্সে উপলব্ধ:

- Ubuntu 12.04, 13.04
- Fedora 19/20+
- RHEL 6.5+
- CentOS 6+
- Gentoo
- ArchLinux
- openSUSE 12.3+
- CRUX 3.0+

65

Docker Container থেকে migrate গেলে আমি কি আমার ডেটা হারাবো?

আপনার Docker Container গুলি সংরক্ষণ করার জন্য আপনার application টি ডিস্কে যে ডেটা লিখেছে তার যে কোনও একটি হিসাবে exit করলে ডেটার কোনও ক্ষতি হয় না। Container টি স্পষ্টভাবে মুছে ফেলা না হওয়া পর্যন্ত এটি করা হবে। Docker Container জন্য ফাইল সিস্টেম Docker Container বন্ধ হওয়ার পরেও টিকে থাকে।

66

কেন Docker ভার্চুয়ালাইজেশন এবং ক্লাউড কম্পিউটিং এর নতুন ক্রেজ?

ভার্চুয়ালাইজেশন এবং ক্লাউড কম্পিউটিং-এর জগতে Docker হল সবথেকে নতুন এবং সর্বশেষ ক্রেজ কারণ এটি একটি অতি-হালকা Containerization অ্যাপ যা এর দক্ষতা প্রমাণ করার সম্ভাবনায় ভরপুর।

67 কেন আমার service গুলি পুনরায় তৈরি করতে বা বন্ধ করতে 10 সেকেন্ড সময় নেয়?

Docker কম্পোজ স্টপ: 10 সেকেন্ড বিলম্ব কারণ এবং সমাধান

কারণ:

- Docker কম্পোজ স্টপ SIGTERM বার্তা পাঠিয়ে Container বন্ধ করে।
- ডিফল্ট টাইমআউট 10 সেকেন্ড।
- 10 সেকেন্ডে Container বন্ধ না হলে, SIGKILL পাঠানো হয়।
- কারণ:
    - CMD-এর JSON form ব্যবহার করা হয় না।
    - SIGTERM-এর জন্য হ্যান্ডলার নেই।

সমাধান:

- CMD-এর JSON form ব্যবহার করুন: ["program", "argument1", "argument2"]
- SIGTERM-এর জন্য হ্যান্ডলার যোগ করুন।
- stop_signal কে application -বোধগম্য সংকেতে সেট করুন।

68 কিভাবে আমি একই হোস্টে একটি কম্পোজ ফাইল একাধিক কপি চালাব?

একই হোস্টে ডকার কম্পোজ ফাইলের একাধিক কপি চালানোর জন্য, আপনি -p ফ্ল্যাগ দিয়ে বিভিন্ন প্রজেক্ট নাম ব্যবহার করতে পারেন। উদাহরণস্বরূপ, আপনি চালাতে পারেন:

```
docker-compose -p project1 up -d
docker-compose -p project2 up -d
```

69 আপ, রান এবং স্টার্ট মধ্যে পার্থক্য কী?

UP:

- docker-compose.yml ফাইল সকল service start/restart শুরু করে।
- "Connected" মোডে: সকল Container লগ দেখা যায়।
- "isolated" মোডে: background Container চালায়।

RUN:

- one-off/ad-hoc কাজের জন্য।
- নির্দিষ্ট service এবং তার Dependentতা চালায়।
- Examination/Administrative কাজ জন্য ব্যবহার করা হয়।
- docker run -ti এর সাথে মিল।

START:

- পূর্বে তৈরি এবং বন্ধ করা Container পুনরায় চালু করে।
- নতুন Container তৈরি করে না।

70

প্রতি হোস্টে কয়টি Container চলতে পারে?

containers সংখ্যা পরিবর্তিত হয় per host run এর available resources উপর ভিত্তি করে যেমন সিপিইউ, মেমরি এবং ডিস্ক স্পেস, সেইসাথে কন্টেইনারের requirements এবং প্ল্যাটফর্মের সীমাবদ্ধতা। এর কারণগুলির উপর নির্ভর করে এটি কয়েক থেকে সম্ভাব্য শত শত পর্যন্ত হতে পারে।

71

Cloud Automation কি শীঘ্রই Containerization কে ছাড়িয়ে যাবে?

এটা অসম্ভাব্য যে Cloud Automation শীঘ্রই কন্টেইনারাইজেশনকে ছাড়িয়ে যাবে। উভয়ই serve distinct purposes, কন্টেইনারাইজেশন অ্যাপ্লিকেশন deployment এবং management এর উপর ফোকাস করে, যখন cloud automation infrastructure managementকে স্ট্রীমলাইন করে। তারা প্রায়ই modern আইটি চাহিদা মেটাতে একসাথে কাজ করে।

72 copy/add or volume সহ একটি নির্দিষ্ট কোড ইনক্লুড করার সম্ভাবনা আছে কি?

হ্যাঁ, আপনি আপনার Docker image বা কন্টেইনারে ফাইল বা ডিরেক্টরিগুলিকে একত্রিত করতে ডকারের copy, add,বা volume কমান্ডের সাথে নির্দিষ্ট কোড ইনক্লুড করতে পারেন।

73 Docker Container অবস্থা সনাক্ত করার একটি উপায় আছে কি?

আমরা 'docker ps –a' কমান্ডটি চালানোর মাধ্যমে একটি Docker Container স্টেটাস সনাক্ত করতে পারি, যা হোস্টে তার সংশ্লিষ্ট স্টেটাস সহ সমস্ত available Docker Container কে লিস্ট ভুক্ত করবে। সেখান থেকে আমরা সহজেই তার স্টেটাস যাচাই করতে আগ্রহ Container টিকে সনাক্ত করতে পারি।

74 'Docker run এবং 'Docker create এর মধ্যে পার্থক্য কী?

‘Docker run এবং Docker create এর মধ্যে পার্থক্য:

Docker ক্রিয়েট:

- stop অবস্থায় একটি Docker Container তৈরি করে।
- Container কে একটি আইডি প্রদান করে।
- পরবর্তী ব্যবহার জন্য আইডি সংরক্ষণ করা যেতে পারে।

Docker রান:

- একটি Container তৈরি এবং চালু করে।
- -cidfile বিকল্প ব্যবহার করে 'docker run' এর সাথে 'docker create' এর কার্যকারিতা অর্জন করা যেতে পারে।

75 একটি Docker Container সময়ে যে কোন সময়ে বিভিন্ন state থাকতে পারে?

একটি Docker Container যে কোনো সময়ে যে কোনো সময়ে থাকতে পারে এমন state নিম্নরূপ দেওয়া হল:

- Created
- Running
- Restarting
- Exited
- Paused
- Dead

76 আপনি Docker থেকে একটি paused Container সরাতে পারেন?

না, Docker থেকে একটি paused Container সরানো সম্ভব নয়। কারণ, Docker Container থেকে রিমুভ করার আগে একটি Container কে বন্ধ (stopped) অবস্থায় থাকতে হয়। কারণ:

- Paused Container কে রিসোর্স বরাদ্দ করা থাকে।
- সরানোর ফলে ডেটা হারাতে পারে।

77

Docker একটি Container নিজে থেকেই পুনরায় চালু করার সম্ভাবনা আছে কি?

ডিফল্ট -রিস্টার্ট ফ্ল্যাগ “no” তে সেট করা থাকে, যার অর্থ Container নিজে থেকে পুনরায় চালু হবে না। তবে, আপনি --restart ফ্ল্যাগ ব্যবহার করে এই behavior পরিবর্তন করতে পারেন।

Ostad

78 কনটেইনারগুলি সরানোর preferred উপায় কী?

Docker থেকে Container সরানোর সর্বোত্তম পদ্ধতি:

'Docker স্টপ' এবং 'Docker আরএম' ব্যবহার করুন:

1."docker stop" ব্যবহার করে Container টি বন্ধ করুন:

- এটি সংকেত পাঠায়।
- application টি বন্ধ করার জন্য সময় দেয়।
- পরিষ্কার কাজগুলি সম্পাদন করে।

2. "docker rm" ব্যবহার করে Container টি সরিয়ে ফেলুন:

- Docker রেজিস্ট্রি আপডেট করে।
- ডিস্ক থেকে ডেটা মুছে ফেলে।

"docker rm -f" ব্যবহার করবেন না:

- এটি SIGKILL সংকেত পাঠায়।
- application টি forcce করে বন্ধ করে।
- ডেটা হারাতে পারে।

79

একটি Container remove করার জন্য সর্বোত্তম পদ্ধতি কোনটি?

প্রথমে Container টি বন্ধ করুন, তারপরে এটি সরান। এখানে কিভাবে:

- $ Docker stop <coontainer_id>
- $ docker rm <container_id>

80

কিভাবে Docker Daemon এবং Docker ক্লায়েন্ট একে অপর সাথে যোগাযোগ করে?

REST API, over UNIX sockets or a network interface এর মাধ্যমে।

81

আপনি কি Docker এ Continuous development (CD) এবং Continuous Integration (CI) বাস্তবায়ন করতে পারেন?

হ্যা, আপনি Docker এ জেনকিন্স চালাতে পারেন এবং ইন্টিগ্রেশন টেস্ট চালানোর জন্য Docker কম্পোজ ব্যবহার করতে পারেন।

82 আপনি কিভাবে একটি Docker Swarm তৈরি করবেন?

কমান্ড ব্যবহার করে :

docker swarm init –advertise-addr <manager IP>

83 আপনি কোথায় Docker ব্যবহার করা হচ্ছে বলে মনে করেন?

Docker ব্যবহার ক্ষেত্র:

কনফিগারেশন Simplification:

- কোডে environment এবং কনফিগারেশন সংরক্ষণ করা যায়।
- সহজে setup করা যায়।

কোড পাইপলাইন ম্যানেজমেন্ট:

- development থেকে production পর্যন্ত ধারাবাহিকতা বজায় রাখে।
- বিভিন্ন সিস্টেমে install করা যায়।

Developer productivity:

- development envirtonment দ্রুত তৈরি করা যায়।
- production কাছাকাছি কাজ করা যায়।

Application আইসোলেশন:

- অ্যাপগুলি বিচ্ছিন্ন থাকে।
- যেকোনো হার্ডওয়্যারে কাজ করতে পারে।

ডিবাগিং ক্ষমতা:

- বিভিন্ন ডিবাগিং tool সাপর্ট করে।
- Container জন্য নির্দিষ্ট ডিবাগিং tool ব্যবহার করা যায়।

মাল্টি-টেন্যান্সি:

- অপ্রয়োজনীয়তা এড়াতে মাল্টি-টেন্যান্ট application রাখা যায়।
- কোড এবং deployment এর দক্ষতা বৃদ্ধি করে।

দ্রুত deployment :

- deployment র সময় হ্রাস করে।
- দ্রুত বাজারে products আনতে সাহায্য করে।

উদাহরণ:

- মাইক্রোসার্ভিসেস-ভিত্তিক application
- ক্লাউড-নেটিভ application
- ডেটা এনালাইসিস
- DevOps
- CI/CD

84

Docker অন্যান্য containerization পদ্ধতি থেকে কিভাবে আলাদা?

Docker অন্যান্য containerization পদ্ধতির তুলনায় অনেক সুবিধা প্রদান করে।

এটি যেকোনো ক্লাউড প্ল্যাটফর্মে সহজে deploy করা যায়, একই হার্ডওয়্যারে আরও application চালানো সম্ভব করে, এবং developer জন্য দ্রুত তৈরি করা সহজ করে তোলে।

Docker অনন্য কনফিগারেশন গুলির মধ্যে রয়েছে image -ভিত্তিক Container, লেয়ারিং, ইউনিয়ন মাউন্ট, ডেটা ভলিউম, নেটওয়ার্কিং, Automatic স্কেলিং এবং Resilience.

এই কনফিগারেশন গুলি Docker কে applicationডেলিভারির জন্য একটি শক্তিশালী এবং ফ্লেক্সিবল প্ল্যাটফর্ম করে তোলে।

85 Docker-এ আমার কম্পোজ ফাইল জন্য আমি কি YAML এর পরিবর্তে JSON ব্যবহার করতে পারি?

আপনি আপনার compose ফাইল জন্য YAML এর পরিবর্তে JSON ব্যবহার করতে পারেন, কম্পোজ সাথে JSON ফাইল ব্যবহার করতে, ব্যবহার করার জন্য JSON ফাইল নাম উল্লেখ করুন, যেমন:

```
$ docker-compose -f docker-compose.json up
```

86

rm কমান্ড ব্যবহার করে সরাসরি Container টি remove করা ভাল নাকি Container টি রিমুভ Container দ্বারা বন্ধ করা ভাল?

Container টি বন্ধ করা এবং তারপর রিমুভ কমান্ড ব্যবহার করে এটি সরিয়ে ফেলা সর্বদা ভাল।

$ docker stop <coontainer_id>

$ docker rm -f <Container_id>

87

Docker এ stateful application চালানো কি একটি ভাল অভ্যাস?

Production জন্য Docker containers স্টেটফুল অ্যাপ্লিকেশন চালানোর জন্য সাধারণত পরামর্শ দেওয়া হয় না। স্টেটফুল ওয়ার্কলোড পরিচালনার জন্য কুবারনেটসের মতো কন্টেইনার অর্কেস্ট্রেশন প্ল্যাটফর্মগুলি ব্যবহার করুন, কারণ তারা একটি containerized environment মধ্যে স্থায়ী স্টোরেজ এবং ডেটা integrity জন্য আরও ভাল অ্যাসিস্ট্যান্স প্রদান করে।

৪৪ ধরুন আপনার কাছে একটি application রয়েছে যার অনেকগুলি Dependent service রয়েছে। Docker Compose কি বর্তমান Container পূর্ববর্তী service Ready হওয়ার জন্য অপেক্ষা করবে?

না, Docker Compose পরবর্তী বা পূর্ববর্তী সার্ভিস Ready হওয়ার জন্যে অপেক্ষা করে না।

শুধু Start হওয়ার জন্যে অপেক্ষা করে।

89 আপনি কিভাবে Production Docker monitoring করবেন?

Production Docker monitoring করার জন্য, আমি Docker Statistics এবং Docker ইভেন্ট ব্যবহার করব। Docker Statistics আমাকে Container CPU এবং মেমরি ব্যবহার সম্পর্কে ধারণা দেবে। Docker ইভেন্টগুলি আমাকে Docker Daemon এ কার্যকলাপ সম্পর্কে তথ্য দেবে।

আমি Prometheus, Grafana, এবং ELK স্ট্যাক মতো tool গুলি ডেটা সংগ্রহ এবং ভিজ্যুয়ালাইজ করার জন্য ব্যবহার করব।

90 প্রোডাকশনে Docker কম্পোজ চালানো কি ভাল অভ্যাস?

হ্যাঁ, Production এ Docker কম্পোজ ব্যবহার করা উচিত। আপনি যখন কম্পোজ সাথে application গুলিকে Define করেন, তখন আপনি এই কম্পোজ definition বিভিন্ন productionপর্যায়ে ব্যবহার করতে পারেন যেমন CI, স্টেজিং, টেস্টিং ইত্যাদি।

91

প্রোডাকশনে নিয়ে যাওয়ার সময় আপনার Docker কম্পোজ ফাইলে কী পরিবর্তন Expected?

প্রোডাকশনে application স্থানান্তর করার সময়, আপনার Docker কম্পোজ ফাইলে নিম্নলিখিত পরিবর্তনগুলি করতে হবে:

ভলিউম বাইন্ডিং:

- সরিয়ে ফেলুন যাতে কোড Container ভিতরে থাকে।
- বাইর পরিবর্তনগুলি রেজিস্টার করে।

পোর্ট বাইন্ডিং:

- হোস্টে বিভিন্ন পোর্টে বাইন্ড করুন।
- applicationঅ্যাক্সেসযোগ্য করার জন্য।

রিজিউম পলিসি:

- নির্দিষ্ট করুন (যেমন, "always" বা "on-failure")।
- Application আপডেট/ব্রাউট ির সময় কন্ট্রোল করে।

অতিরিক্ত service:

- লগ এগ্রিগেটর যোগ করুন।
- central লগিং এবং পর্যবেক্ষণ জন্য।

92

আপনি কি Kubernetes ব্যবহার করেছেন? আপনার যদি থাকে তবে আপনি Docker Swarm এবং Kubernetes মধ্যে কোনটিকে পছন্দ করবেন?

হ্যাঁ, আমি Kubernetes ব্যবহার করেছি।

Kubernetes-এর সুবিধা:

- complex application স্কেল করতে সহজ।
- বৃদ্ধি নিরাপত্তা কনফিগারেশন ।
- বিভিন্ন ধরণ applicationজন্য উপযুক্ত।

Docker Swarm-এর সুবিধা:

- দ্রুত সেটআপ এবং ডেপ্লয়মেন্ট।
- ছোট টিম জন্য ব্যবহার করা সহজ।
- কম রিসোর্স ব্যবহার করে।

আমার প্রয়োজন জন্য, Kubernetes-এর বৃদ্ধি স্কেলিং এবং নিরাপত্তা কনফিগারেশন গুলি আমার জন্য গুরুত্বপূর্ণ।

Docker প্রতিযোগীদ মধ্যে, আমি Nomad এবং Mesos সম্পর্কে জানি।

93 আপনি কি container এবং হোস্ট জুড়ে লোড ব্যালেন্সিং সম্পর্কে জানেন ? এটা কিভাবে কাজ করে?

হ্যাঁ, আমি Container এবং হোস্ট জুড়ে লোড ব্যালেন্সিং সম্পর্কে জানি। লোড ব্যালেন্সিং হল একাধিক হোস্ট জুড়ে Container ট্রাফিক distribution করার process। এটি সিস্টেম স্কেলিং এবং performance বৃদ্ধি করতে পারে।

HAProxy একটি জনপ্রিয় ওপেন সোর্স লোড ব্যালেন্সার যা বিভিন্ন Healthy Container ট্রাফিক distribution করতে পারে। যদি একটি Container crash হয়, HAProxy Automatic ভাবে ট্রাফিক একটি নতুন রানিং Container পুনরায় রাউট করবে।

Container জন্য লোড ব্যালেন্সিং এবং HAProxy গুরুত্বপূর্ণ কারণ তারা সিস্টেম স্কেলিং, performance এবং হাই এভেইল্যাবিলিটি বাড়াতে পারে।

94 ক্লায়েন্ট কি?

Docker ক্লায়েন্ট Docker ডেমন সাথে যোগাযোগ জন্য একটি CLI। এটি image তৈরি, রান, বন্ধ, ম্যানেজ করার জন্য ব্যবহৃত হয়।

95 Docker ট্রাস্টেড রেজিস্ট্রি ব্যাখ্যা করুন?

Docker ট্রাস্টেড রেজিস্ট্রি একটি নিরাপদ image স্টোরেজ টোল যা ডেটা সেন্টারে application deployment জন্য ব্যবহার করা যেতে পারে। এটি firewall এর পেছনে ইনস্টল করা উচিত এবং application ব্যবহৃত Docker image গুলিকে কন্ট্রোল করতে সাহায্য করে।

96

Docker অধীনে ডিফল্ট লগিং ড্রাইভার কীভাবে কনফিগার করবেন?

একটি নির্দিষ্ট লগিং ড্রাইভারে ডিফল্ট হিসাবে Docker ডেমন কনফিগার করতে। আপনাকে লগ-ড্রাইভার মান deemon.jason.fie লগিং ড্রাইভ নাম সাথে সেট করতে হবে।

97

Docker এ বিভিন্ন ধরণ ভলিউম মাউন্ট কি কি পাওয়া যায়?

ভলিউম মাউন্ট, বাইন্ড মাউন্ট এবং tmpfs মাউন্ট।

98

Docker কি IPV6 এর জন্য সাপর্ট দেয়?

হ্যাঁ, Docker IPv6 সাপর্ট প্রদান করে। IPv6 নেটওয়ার্কিং শুধুমাত্র Linux হোস্টে চালানো Docker Daemon এ সাপর্টেড । আপনি যদি Docker ডেমনে IPv6 সাপর্ট করাতে চান, তাহলে আপনাকে /etc/docker/daemon.json পরিবর্তন করতে হবে এবং ipv6 keyটিকে true সেট করতে হবে।

99

CNM কি?

CNM (Container Networking Model) Docker, Inc. দ্বারা তৈরি একটি স্ট্যান্ডার্ড বা স্পেসিফিকেশন যা Docker environment Container নেটওয়ার্কিং-এর জন্য ভিত্তি তৈরি করে।

Ostad

100 K8s কি?

K8s হল Kubernetes-এর আরেকটি শব্দ।

101

Software এবং DevOps এর ক্ষেত্রে Orchestration কি?

Orchestration হল একাধিক service একীকরণ যা তাদের process গুলি Automatic করে এবং সময়মত তথ্য সিঙ্ক্রোনাইজ করে।

102 Docker Swarm এবং Kubernetes এর মধ্যে প্রধান পার্থক্যগুলি কী কী?

Docker Swarm বনাম Kubernetes: মূল পার্থক্য

সেট আপ:

- Docker Swarm: সহজ
- Kubernetes: জটিল

ক্লাস্টার:

- Docker Swarm: দুর্বল
- Kubernetes: শক্তিশালী

অটো-স্কেলিং:

- Docker Swarm: সাপর্ট করে না
- Kubernetes: সাপর্ট করে

GUI:

- Docker Swarm: নেই
- Kubernetes: আছে

লোড ব্যালেন্সিং:

- Docker Swarm: Automatic
- Kubernetes: ম্যানুয়াল

লগিং এবং মনিটরিং:

- Docker Swarm: 3rd Party Tool
- Kubernetes: Integrated Tool

স্টোরেজ ভলিউম:

- Docker Swarm: যেকোনো Container শেয়ার করা যায়
- Kubernetes: একই পড Container শেয়ার করা যায়

রোলিং আপডেট:

- Docker Swarm: সাপর্ট করে
- Kubernetes: সাপর্ট করে

Automatic রোলব্যাক:

- Docker Swarm: সাপর্ট করে না
- Kubernetes: সাপর্ট করে

103

Kubernetes architecture এর প্রধান component কি কি?

Kubernetes আর্কিটেকচার দুটি প্রাথমিক component রয়েছে: মাস্টার নোড এবং ওয়ার্কার নোড।

104 kube-apiserver ভূমিকা কী?

kube-apiserver ভূমিকা:

- API অবজেক্ট জন্য কনফিগারেশন ডেটা যাচাই এবং প্রদান করে।
- Pod, Service, Replication Controller ইত্যাদি পরিচালনা করে।
- REST অপারেশন এবং ক্লাস্টার ফ্রন্টএন্ড প্রদান করে।
- ক্লাস্টার স্টেট শেয়ার করে যার মাধ্যমে অন্য সব কম্পোনেন্ট ইন্টারঅ্যাক্ট করে।

105 Kubernetes A নোড কি?

A node হলো Kubernetes এর একটি worker মেশিন যেখানে কন্টেনার deploy করা হয়। এটি অ্যাপ্লিকেশন চালানোর দায়িত্ব পালন করে এবং CPU, মেমোরি, এবং নেটওয়ার্কিং সরবরাহ করার জন্য প্রয়োজনীয় resource গণনা করে।

106 নোড স্ট্যাটাস কি ধারণ করে?

একটি নোড স্ট্যাটাস প্রধান component হল address, শর্ত, ক্ষমতা এবং তথ্য।

107 Kubernetes মাস্টার নোডে কোন process চলে?

Kube-api সার্ভার process টি মাস্টার নোডে চলে এবং deploy স্কেল করে।

108 Kube-scheduler কাজ কী?

Kube-scheduler নতুন তৈরি পডগুলিতে নোডগুলি allocate করে।

109 Kubernetes Container ক্লাস্টার কী?

Container একটি ক্লাস্টার হল মেশিন component গুলির একটি সেট যা নোড। ক্লাস্টারগুলি নির্দিষ্ট রাউট শুরু করে যাতে নোডগুলিতে রানিং Container গুলি একে অপর সাথে যোগাযোগ করতে পারে। Kubernetes-এ, Container ইঞ্জিন (Kubernetes API-এর সার্ভার নয়) API সার্ভার জন্য হোস্টিং প্রদান করে।

110 Google Container Engine কি?

Google Container Engine হল একটি ওপেন-সোর্স ম্যানেজমেন্ট প্ল্যাটফর্ম যা Google পাবলিক ক্লাউড service গুলিতে চলা ক্লাস্টারগুলির জন্য সাপর্ট প্রদান করার জন্য Docker Container এবং ক্লাস্টারগুলির জন্য তৈরি।

111 Kubernetes 'হিপস্টার' কী?

হিপস্টার হল কুবলেট দ্বারা সংগৃহীত ডেটার জন্য একটি performance পর্যবেক্ষণ এবং মেট্রিক্স সংগ্রহ সিস্টেম।

112 Minikube কি?

Local ভাবে Kubernetes ব্যবহার করার সহায়তা করে। Windows, macOS এবং Linux PC তে সিঙ্গেল-নোড Kubernetes ক্লাস্টার চালাতে দেয়।

113 Kubernetes নেমস্পেস কি?

একাধিক ব্যবহারকারীর মধ্যে ক্লাস্টার resources ভাগ করার জন্য নেমস্পেস ব্যবহার করা হয়।

114 প্রাথমিক নেমস্পেস নাম বলুন যেখান থেকে Kubernetes শুরু হয়?

ডিফল্ট

- Kubernetes-এর ডিফল্ট নেমস্পেস "default"।
- নতুন deployments, pods, services ইত্যাদি এই নেমস্পেসে তৈরি হয় যদি না অন্যটি উল্লেখ করা হয়।

115 Kubernetes কন্ট্রোলার ম্যানেজার কি?

কন্ট্রোলার ম্যানেজার Kubernetes-এর একটি গুরুত্বপূর্ণ Daemon। এটি Replication Controller, Node Controller, Service Controller ইত্যাদি বিভিন্ন কন্ট্রোলার ব্যবহার করে ক্লাস্টার স্টেটাস কন্ট্রোল করে। কন্ট্রোলার API Server-এর মাধ্যমে ক্লাস্টার সাথে যোগাযোগ করে এবং ক্লাস্টার কাঙ্ক্ষিত অবস্থার সাথে বর্তমান অবস্থা মেলে। এটি অব্যবহৃত রিসোর্স মুছে ফেলে এবং নেমস্পেস তৈরি করে।

116 কন্ট্রোলার ম্যানেজার কত প্রকার?

মাস্টার নোডে রানিং প্রাথমিক নিয়ামক পরিচালক:

- এন্ডপয়েন্ট কন্ট্রোলার
- সার্ভিস অ্যাকাউন্ট কন্ট্রোলার
- নেমস্পেস কন্ট্রোলার
- নোড কন্ট্রোলার
- টোকেন কন্ট্রোলার
- রেপ্লিকেশন কন্ট্রোলার

117 etcd কি?

etcd Kubernetes-এর জন্য একটি গুরুত্বপূর্ণ ডেটা স্টোর। এটি মেটাডেটা এবং কনফিগারেশন ডেটা সংরক্ষণ করে এবং ক্লাস্টার নোডগুলিকে ডেটা অ্যাক্সেস করার অনুমতি দেয়। etcd ওপেন-সোর্স এবং বিভিন্ন OS-এ কাজ করে, যা এটিকে Kubernetes-এর জন্য একটি বহুমুখী এবং স্টেটাস শীল সমাধান করে তোলে।

118 Kubernetes মধ্যে বিভিন্ন service কী কী?

Kubernetes serviceeগুলির বিভিন্ন ধরণ ইনক্লুড :

- ক্লাস্টার আইপি service
- নোড পোর্ট service
- External name creation service এবং
- লোড ব্যালেন্সার service

119 ClusterIP কী?

ClusterIP হল ডিফল্ট Kubernetes service যা একটি ক্লাস্টার ভিতরে একটি service প্রদান করে (কোন বাহ্যিক অ্যাক্সেস ছাড়াই) যা আপনার ক্লাস্টার মধ্যে থাকা অন্যান্য অ্যাপগুলি অ্যাক্সেস করতে পারে।

120 NodePort কী?

নোডপোর্ট service টি আপনার service এ সরাসরি External ট্র্যাফিক পাওয়ার সবচেয়ে Basic way।

এটি সমস্ত নোডে একটি নির্দিষ্ট পোর্ট খোলে এবং এই পোর্টে প্রিত যেকোন ট্র্যাফিককে সার্ভিসে ফরওয়ার্ড করে।

121 Kubernetes লোডব্যালেন্সার কী?

লোডব্যালেন্সার service টি ইন্টারনেটে service গুলি প্রকাশ করতে ব্যবহৃত হয়। একটি নেটওয়ার্ক লোড ব্যালেন্সার, উদাহরণস্বরূপ, একটি সিঙ্গেল IP ঠিকানা তৈরি করে যা আপনার serviceতে সমস্ত ট্র্যাফিক ফরোয়ার্ড করে।

122 Container রিসোর্স মনিটরিং কী?

এটি এমন activity কে বোঝায় যা মেট্রিক্স সংগ্রহ করে এবং Containerized applicationএবং মাইক্রোসার্ভিস Environment ট্র্যাক করে। এটি performance বৃদ্ধি করতে সাহায্য করে এবং নিশ্চিত করে যে তারা smoothly কাজ করে।

123 একটি headless service কী?

একটি হেডলেস service একটি ClusterIP এর সাথে আবদ্ধ না হয়ে service আবিষ্কার process গুলির সাথে ইন্টারফেস করতে ব্যবহৃত হয়, তাই আপনাকে প্রক্সির মাধ্যমে অ্যাক্সেস না করেই সরাসরি পডগুলিতে পৌঁছানোর অনুমতি দেয়। যখন লোড ব্যালেন্সিং বা সিঙ্গেল service আইপি প্রয়োজন হয় না তখন এটি কার্যকর।

124 ফেডারেটেড ক্লাস্টার কী?

একাধিক ক্লাস্টার একত্রীকরণ যা একটি সিঙ্গেল লজিক্যাল ক্লাস্টার হিসাবে বিবেচনা করে ক্লাস্টার ফেডারেশনকে বোঝায়। এতে, একাধিক ক্লাস্টার সিঙ্গেল ক্লাস্টার হিসাবে পরিচালিত হতে পারে। তারা ফেডারেটেড গ্রুপ সহায়তায় থাকে। এছাড়াও, ব্যবহারকারীরা ডেটা সেন্টার বা ক্লাউড মধ্যে বিভিন্ন ক্লাস্টার তৈরি করতে পারে এবং সেগুলিকে এক জায়গায় কন্ট্রোল বা পরিচালনা করতে ফেডারেশন ব্যবহার করতে পারে। আপনি নিম্নলিখিতগুলি করে ক্লাস্টার ফেডারেশন সম্পাদন করতে পারেন:

- ক্রস ক্লাস্টার যা অংশগ্রহণকারী ক্লাস্টার থেকে ব্যাকএন্ড সহ DNS এবং লোড ব্যালেন্সার থাকার ক্ষমতা প্রদান করে।
- ব্যবহারকারীরা বিভিন্ন ক্লাস্টার জুড়ে একই deployment এর সেট deploy করার জন্য বিভিন্ন ক্লাস্টার জুড়ে resources গুলি সিঙ্ক করতে পারে।

125 Kubectl কী?

Kubectl হল একটি service এজেন্ট যেটি Kubernetes API সার্ভার মাধ্যমে পড স্পেস দেখার মাধ্যমে পডগুলির একটি সেট কন্ট্রোল ও রক্ষণাবেক্ষণ করে। এটি নিশ্চিত করে পড Life-cycle সংরক্ষণ করে যে প্রদত্ত Container গুলির হিসাবে চলছে। Kubectl প্রতিটি নোডে চলে এবং মাস্টার এবং স্লেভ নোড মধ্যে যোগাযোগ সক্ষম করে।

126 Kubectl কী?

Kubectl হল একটি CLI (কমান্ড-লাইন ইন্টারফেস) যা Kubernetes ক্লাস্টারগুলির জন্যে কমান্ড চালানোর জন্য ব্যবহৃত হয়। যেমন, এটি Kubernetes কম্পোনেন্টে বিভিন্ন ক্রিয়েট এবং ম্যানেজ কমান্ড মাধ্যমে Kubernetes ক্লাস্টার ম্যানেজারকে কন্ট্রোল করে।

127 Kubernetes জন্য Recommended নিরাপত্তা ব্যবস্থার উদাহরণ দিন।

স্ট্যান্ডার্ড Kubernetes নিরাপত্তা ব্যবস্থার উদাহরণ মধ্যে রয়েছে রিসোর্স কোটা Define করা, নিরীক্ষার জন্য সাপর্ট, etcd অ্যাক্সেস সীমাবদ্ধতা, environment নিয়মিত নিরাপত্তা আপডেট, নেটওয়ার্ক বিভাজন, কঠোর resource নীতির definition, নিরাপত্তা দুর্বলতার জন্য Continuous স্ক্যান করা এবং অনুমোদিত রিপোজিটরি থেকে image ব্যবহার করা।

128 Kube-proxy কী?

Kube-proxy হল একটি লোড ব্যালেন্সার এবং নেটওয়ার্ক প্রক্সির বাস্তবায়ন যা অন্যান্য নেটওয়ার্কিং একটিভিটি এর সাথে service abstraction সাপর্ট করতে ব্যবহৃত হয়। Kube-proxy আইপি এবং আগত Request পোর্ট নম্বর উপর ভিত্তি করে সঠিক Container ট্রাফিককে নির্দেশ করে।

129 আপনি কিভাবে একটি Kubernetes লোড ব্যালেন্সার জন্য একটি static আইপি পেতে পারেন?

Kubernetes লোড ব্যালেন্সার জন্য একটি static আইপি ডিএনএস রেকর্ড পরিবর্তন করে পাওয়া যেতে পারে যেহেতু Kubernetes মাস্টার একটি নতুন static আইপি ঠিকানা বরাদ্দ করতে পারে।

130 Ingress network কি এবং এটি কিভাবে কাজ করে?

Ingress Network Kubernetes-এর একটি গুরুত্বপূর্ণ কনফিগারেশন যা ব্যবহারকারী ক্লাস্টার বাইরে থেকে service গুলিতে অ্যাক্সেস করতে দেয়। এটি নিয়ম তৈরি করে অ্যাক্সেস কন্ট্রোল করা সহজ করে এবং লোড ব্যালেন্সার প্রয়োজনীয়তা দূর করে।

কীভাবে কাজ করে:

- একটি API অবজেক্ট যা HTTPS/HTTP এর মাধ্যমে service গুলিতে External অ্যাক্সেস পরিচালনা করে।
- ট্র্যাফিক রাউট করার নিয়ম সেট আপ করা সহজ করে।
- লোড ব্যালেন্সার বা প্রতিটি নোডে service প্রকাশ প্রয়োজন নেই।

131 PDB (Pod Disruption Budget) কী?

PDB Kubernetes-এর একটি গুরুত্বপূর্ণ কনফিগারেশন যা application HA নিশ্চিত করে। minAvailable কনফিগারেশন ব্যবহার করে, আমরা নির্দিষ্ট করতে পারি যে একটি নির্দিষ্ট সংখ্যক পড সবসময় রানিং থাকবে। এটি নিশ্চিত করে যে application সর্বদা ব্যবহারকারী কাছে available থাকে।

উদাহরণ: minAvailable => ব্যবহার করে YAML কনফিগারেশন

```
apiVersion: policy/v1beta1
kind: PodDisruptionBudget
metadata:
 name: zk-pdb
spec:
 minAvailable: 2
 selector:
  matchLabels:
    app: zookeeper
```

132 init Container কী এবং কখন এটি ব্যবহার করা যেতে পারে?

init Container গুলি POD-এর জন্য প্রস্তুতিমূলক কাজগুলি সম্পাদন করে। application Container শুরু করার আগে ডেটাবেস সংযোগ deploy, ডেটা কপি করা, service গুলি শুরু করা এবং এনভায়রনমেন্ট ভ্যারিয়েবল সেট করার জন্য init container ব্যবহার করা যেতে পারে।

উদাহরণস্বরূপ, একটি init container sleep 60 কমান্ড ব্যবহার করে application শুরু করার আগে 60 সেকেন্ড অপেক্ষা করতে পারে। এটি ডেটাবেস মতো নির্ভরতাগুলি চালু এবং স্টেটাস রানিং হওয়ার জন্য সময় দেয়।

133 Kubernetes নিরাপত্তা বাড়ানোর জন্য বিভিন্ন জিনিস কী করা যেতে পারে?

ডিফল্টরূপে, POD অন্য যেকোনো POD এর সাথে যোগাযোগ করতে পারে, আমরা POD-এর মধ্যে এই যোগাযোগ সীমিত করতে নেটওয়ার্ক নীতি সেট আপ করতে পারি।

- RBAC (Role-based access control) অনুমতি limited করতে।
- safety margin deploy করতে নেমস্পেস ব্যবহার করুন।
- সুবিধাপ্রাপ্ত Container চালানো এড়াতে Admission Control Policy সেট করুন।
- অডিট লগিং চালু করুন।

134 POD থেকে central লগ কিভাবে পেতে হয়?

এই আর্কিটেকচার প্রয়োগ এবং অন্যান্য অনেক কারণ উপর নির্ভর করে। নিম্নলিখিত সাধারণ লগিং নিদর্শন আছে:

- নোড লেয়ার লগিং এজেন্ট।
- স্ট্রিমিং সাই Docker Container।
- লগিং এজেন্ট সহ সাইDocker Container।
- applicationথেকে সরাসরি লগ রপ্তানি করুন।

সেটআপে, জার্নালবিট এবং ফাইলবিট ডেমনসেট হিসাবে চলছে।

135 কিভাবে TLS Ingress সাথে কনফিগার করা উচিত?

tls এবং secretName এন্ট্রি যোগ করুন।

```
spec:
 tls:
 - hosts:
   - some_app.com
   secretName: someapp-secret-tls
```

Ostad

136 নিম্নলিখিত ফাইলে কোন service এবং নেমস্পেস উল্লেখ করা হয়েছে?

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: some-configmap
data:
  some_url: silicon.chip
```

উত্তরঃ উপর ফাইল থেকে এটা স্পষ্ট যে "সিলিকন" serviceটি "চিপ" নামক একটি নেমস্পেস একটি রেফারেন্স

137 Operator কী?

operator গুলি Kubernetes এক্সটেনশন হিসাবে কাজ করে এবং application এবং তার component গুলি পরিচালনা করার ক্ষমতা প্রদান করে। এদের দায়িত্ব হলো সাধারণত Kubernetes সম্পর্কিত নীতি মেনে চলা, বিশেষ করে কন্ট্রোল লুপগুলির সাথে সম্পর্কিত।

138 Operator এর উদ্দেশ্য কী?

অপারেটরের উদ্দেশ্য হল কুবারনেটিস ক্লাস্টারে অ্যাপ্লিকেশনগুলির পরিচালনা এবং অপারেশনগুলির অটোমেশন করা।

139 GKE কী?

GKE হল Google Kubernetes Engine যা Docker Container গুলির জন্য সিস্টেম পরিচালনা এবং অর্কেস্ট্রেট করার জন্য ব্যবহৃত হয়। Google পাবলিক ক্লাউড সাহায্যে, আমরা Container ক্লাস্টারও অর্কেস্ট্রেট করতে পারি।

140 Ingress ডিফল্ট ব্যাকএন্ড কী?

এটি সুনির্দিষ্ট করে যে Kubernetes ক্লাস্টারে একটি ইনকামিং Request সাথে কী করতে হবে যা কোনো ব্যাকএন্ডে ম্যাপ করা হয় না অর্থাৎ ইনকামিং HTTP Request জন্য কোনো নিয়ম Defined না হলে কী করতে হবে যদি ডিফল্ট ব্যাকএন্ড serviceটি Define করা না থাকে, তাহলে এটিকে Define করার পরামর্শ দেওয়া হয়।

141 Local ভাবে Kubernetes কিভাবে চালাবেন?

Local Kubernetes চালানোর দুটি প্রধান উপায় রয়েছে:

১. Minikube ব্যবহার করে:

- Minikube হল একটি tool যা আপনাকে আপনার কম্পিউটারে একটি VM-এ সিঙ্গেল -নোড ক্লাস্টার চালাতে দেয়।
- এটি Kubernetes শেখার জন্য একটি দুর্দান্ত উপায় কারণ এটি সেট আপ করা সহজ এবং ব্যবহার করা সহজ।
- Minikube ব্যবহার করে Localভাবে Kubernetes চালানোর জন্য, আপনাকে প্রথমে Minikube ইনস্টল করতে হবে।
- Minikube ইনস্টল করার পরে, আপনি "minikube start" কমান্ডটি ব্যবহার করে একটি ক্লাস্টার শুরু করতে পারেন।
- ক্লাস্টার শুরু হয়ে গেলে, আপনি "kubectl" কমান্ডটি ব্যবহার করে এটির সাথে ইন্টারঅ্যাক্ট করতে পারেন।

২. Docker ব্যবহার করে:

- Docker হল একটি প্ল্যাটফর্ম যা আপনাকে application গুলিকে Container প্যাকেজ করতে এবং চালাতে দেয়।
- আপনি Docker ব্যবহার করে Kubernetes ক্লাস্টার সমস্ত component চালাতে পারেন।
- Docker ব্যবহার করে Localভাবে Kubernetes চালানোর জন্য, আপনাকে প্রথমে Docker ইনস্টল করতে হবে।
- Docker ইনস্টল করার পরে, আপনি "docker run" কমান্ডটি ব্যবহার করে Kubernetes নোড চালাতে পারেন।
- আপনি "kubectl" কমান্ডটি ব্যবহার করে ক্লাস্টার সাথে ইন্টারঅ্যাক্ট করতে পারেন।

142 ব্যাখ্যা করুনঃ

```
spec:
  Containers:
    - name: USER_PASSWORD
      valueFrom:
        secretKeyRef:
          name: some-secret
          key: password
```

উত্তরঃ ব্যাখ্যা-

USER_PASSWORD environment variable "কিছু-সিক্রেট" নামক গোপনে পাসওয়ার্ড key থেকে মান সংরক্ষণ করবে।

143 POD নির্ধারিত না হলে কিভাবে সমস্যা সমাধান করবেন?

"K8 pod গুলোকে নোডগুলিতে তৈরি করার জন্য দায়ী। শুরু না হওয়া pod এর অনেক কারণ থাকতে পারে। সবচেয়ে সাধারণ কারণ হল রিসোর্স অভাব (resource shortage)।

কিভাবে সমস্যা সমাধান করবেন:

- kubectl describe <POD> -n <Namespace> কমান্ড ব্যবহার করে POD শুরু না হওয়ার কারণ ।
- kubectl get events কমান্ড ব্যবহার করে ক্লাস্টার থেকে আসা সমস্ত ইভেন্ট দেখুন।
- kubectl logs <POD> -n <Namespace> কমান্ড ব্যবহার করে POD এর logs দেখুন।
- kubectl top nodes কমান্ড ব্যবহার করে নোডগুলিতে রিসোর্স ব্যবহার টেস্ট করুন।

কিছু সম্ভাব্য সমাধান:

- Resource quotas বাড়ান।
- Pod limits কমিয়ে দিন।
- Nodes যোগ করুন।
- Pods কে অন্য nodes এ reschedule করুন।

144 K8 এ বাহ্যিক নেটওয়ার্ক সংযোগ প্রদান বিভিন্ন উপায় কি কি?

ডিফল্টরূপে, POD বাহ্যিক নেটওয়ার্কে পৌঁছাতে সক্ষম হওয়া উচিত কিন্তু এর বিপরীতে কিছু পরিবর্তন করতে হবে। outside থেকে POD এর সাথে সংযোগ করার জন্য নিম্নলিখিত বিকল্পগুলি available:

- নোডপোর্ট (এটি প্রতিটি নোড সাথে যোগাযোগ করতে একটি পোর্ট প্রকাশ করবে)
- লোড ব্যালেন্সার (TCP/IP প্রোটোকল L4 লেয়ার)
- Entry (TCP/IP প্রোটোকল L7 লেয়ার)

আরেকটি পদ্ধতি হল Kube-proxy ব্যবহার করা যা Local সিস্টেম পোর্টে শুধুমাত্র ক্লাস্টার আইপি সহ একটি service প্রকাশ করতে পারে।

145 আমরা কিভাবে পোর্ট ফরওয়ার্ড করতে পারি '8080 (Container) -> 8080 (সার্ভিস) -> 8080 (Ingress) -> 80 (ব্রাউজার) এবং এটি কীভাবে করা যায়?

এই পোর্ট ফরওয়ার্ডিং process টি সম্পন্ন করতে, আমাদের তিনটি ধাপ অনুসরণ করতে হবে:

ধাপ ১: Ingress কন্ট্রোলার তৈরি করা

প্রথমে, আমাদের একটি "Ingress কন্ট্রোলার" তৈরি করতে হবে। এটি একটি পড যা external ট্র্যাফিক গ্রহণ করে এবং Ingressকে পরিচালনা করে। Ingress কন্ট্রোলার তৈরি করার জন্য, আমরা নিম্নলিখিত কমান্ডটি ব্যবহার করতে পারি:

```
kubectl apply -f ingress-controller.yaml
```

ধাপ ২: Ingress তৈরি করা

এবার, আমাদের একটি "Ingress" তৈরি করতে হবে। Ingress হল একটি resource যা ট্র্যাফিক নিয়ম নির্ধারণ করে এবং

এটিকে বিভিন্ন service মধ্যে রাউট করে। Ingress তৈরি করার জন্য, আমরা নিম্নলিখিত কমান্ডটি ব্যবহার করতে পারি:

```
kubectl apply -f ingress.yaml
```

ধাপ ৩: Ingress সিলেক্টর তৈরি করা

অবশেষে, আমাদের একটি "Ingress সিলেক্টর" তৈরি করতে হবে। Ingress সিলেক্টর নির্ধারণ করে যে কোন Ingress কন্ট্রোলার কোন Ingress পরিচালনা করবে। Ingress সিলেক্টর তৈরি করার জন্য, আমরা নিম্নলিখিত কমান্ডটি ব্যবহার করতে পারি:

```
kubectl apply -f ingress-selector.yaml
```

এই তিনটি ধাপ সম্পন্ন করার পর, পোর্ট ফরওয়ার্ডিং process টি সম্পন্ন হবে।

146 Container Orchestration কী?

Container Orchestration হলো একাধিক Containerized application কে একসাথে পরিচালনা করার process। এটি একটি application বিভিন্ন অংশকে স্বাধীনভাবে স্কেল এবং পরিচালনা করতে সাহায্য করে।

147 Container Orchestration এর প্রয়োজন কী?

Container অর্কেস্ট্রেশনের প্রয়োজন হলো কন্টেনারাইজড অ্যাপ্লিকেশনগুলির অটোমেটিক ডিপ্লয়মেন্ট, স্কেলিং, এবং ম্যানেজমেন্ট। এটি নিশ্চিত করে যে কন্টেনারগুলি প্রয়োজনীয় হোস্টগুলিতে দ্রুততার সাথে distributed হয়, উচ্চ availability বজায় রাখে, এবং ডিমান্ডের উপর ভিত্তি করে সহজেই স্কেল আপ বা ডাউন করা যায়।

148 Heapster কী?

Heapster হল একটি Cluster-wide aggregator যা প্রতিটি নোডে রানিং Kubelet দ্বারা প্রদত্ত ডেটা সংগ্রহ করে। এটি CPU, memory, disk usage, network traffic ইত্যাদির মতো মেট্রিক সংগ্রহ করে এবং InfluxDB, Grafana, Prometheus-এর মতো ডেটা স্টোরেজ সিস্টেমে পাঠায়।

149

Kubernetes Architecture এর বিভিন্ন component কী কী?

Kubernetes Architecture দুটি প্রধান component:

মাস্টার নোড:

Kubernetes কন্ট্রোল প্লেন:

- Kube-APIServer: API endpoints প্রদান করে।
- Kube-Controller-Manager: ক্লাস্টার অবস্থা কন্ট্রোল করে।
- Kube-Scheduler: পডগুলিকে ওয়ার্কার নোডগুলিতে নির্ধারণ করে।

ওয়ার্কার নোড:

- Kubelet: ক্লাস্টার সাথে ওয়ার্কার নোডকে সংযুক্ত করে।
- Kube-Proxy: সার্ভিস জন্য লোড ব্যালান্সিং প্রদান করে।