Spring And Hibernate

Sunday, August 8, 2021 2:04 PM

Spring Development process

- Configure your spring beans
- Create a spring Container
- Retrieve beans from spring container

Spring Container is generally known as **ApplicationContext**

Spring Container

- Primary functions
  - Create and manage objects(Inversion of Control)
  - Inject object dependencies(Dependency Injection)

What is Spring Bean?

A spring bean is simply a java object. When java objects are created by the spring container then spring refers to them as Spring beans.

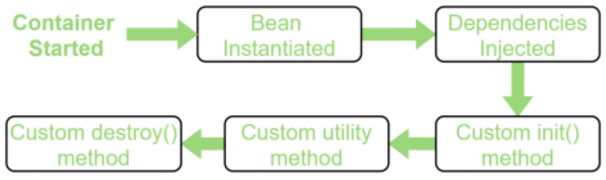
Spring Dependency Injection

The client delegates to call to another object the responsibility of providing its dependencies.

Bean Scope

- Scope refers to the lifecycle of a bean.
- How long does the bean live?
- How many instances are created?
- How is the bean shared?
- Default scope is singleton.
  - What is Singleton?
    - Spring container creates only one instance of the bean by default.
    - It is cached in memory
    - All requests for the bean will return a shared reference to the same bean.
    - Coach thecoach = context.getBean("mycoach", Coach.class) **and**
    - Coach thecoach = context.getBean("mycoach", Coach.class) give you the reference of same bean. Same area of memory and same reference.
-

Scope	Descriptions
Singleton	Create a single shared instance of the bean. Default Scope
Prototype	Create a new bean instance for each container request.
Request	Scoped to an HTTP web request. Only used for web apps.
Session	Scoped to an HTTP web session. Only used for web apps.
Global-session	Scoped to a global HTTP web session. Only used for web apps.

- Bean Lifecycle
  - 

Annotation: give the metadata of the class.

- Eg @override
- Xml configuration can be verbose
- Annotations minimizes the XML configurations
- **Scanning for your component class**
- Spring will scan your java classes for special annotations
- Automatically register the beans in the spring container
- **Development Process for annotation enables**
  - Enable the component scanning in Spring config file ==> **<bean> <context:component-scan base-pacakge="com.luv2Code.springDemo" /> </bean>**
  - Add the @Component Annotation to your java class --> Register the spring bean automatically.
  - Retrieve bean from the Spring container --> context.getBean("beanName");

What is spring AutoWiring?

- For Dependency Injection spring can use auto wiring
- Spring will look for the class that matches the property
  - Matches by type: Class or interface

- Spring will inject it automatically .. Hence it is autowired

    - **Constructor Injection**
    - **Setter injection**
        - **Any Method on your class**
    - **<u>Field injection (Mainly use)</u>**
        - Inject dependencies by setting field values on your class directly -- even private fields
            - (**Accomplished by using Java Reflection**)
- Which injection type should you use?

Azure/Devops

Sunday, August 8, 2021 5:46 PM

What is cloud computing?

Cloud computing is basically a great level of abstraction over the infrastructure that can help you to focus more on your business logic without having to worry about hosting or infrastructure needs. This is the general term used for delivering the hosted services over the internet. In cloud computing the computing resources are providing "as a service". Just like Electricity, you pay to electricity provider and get the electricity at your home and for this you don't need to create any infrastructure. Cloud computing provides you computing resources as a service where you don't have to focus on infrastructure requirements.

What are the benefits, advantages of cloud computing?

- Scalability
- Agility
- High Availability
- Pay as you go
- Moving from Capex to Opex
- Fault Tolerance
- High Response Time
- High Bandwidth
- Low Latency

App service Features:

- Scaling
- Security and compliance

Why Cloud?

Cloud is becoming the new standard for business computing and delivery of software and applications, and is rapidly overtaking the traditional in-house system as a **reliable, scalable, innovative, secure and cost-effective** IT Platform.

- Create it in the cloud within minutes
- Use it as you wish
- Pay for what you use
- Shut it down when not needed
- Automatically maintained patched secured monitored

==> Characteristics of Cloud

- On-Demand self service
- Broad Network Access
- Resource Pooling
    - Physical resources are shared between customers
- Rapid Elasticity
    - Resources can be scaled up and down as needed automatically
- Measured Services
    - Payment is done only for resources actually used
    - Server time/ Db storage/ function call

CapEX(capital Expenses) VS OpEx(Operating Expenses)

IaaS-

- Infrastructure as a service
- The cloud provides the underlying platform
    - Compute/ Networking /storage
- The client handles and is responsible for all the rest.
- Ex - Virtual machine

PaaS -

- The cloud provides platform for running apps
- Compute/ Networking/ storage/ runtime environment / Scaling/ Redundancy / updates/ patching
- Ex web apps
- Lift and shift capability
- Turnkey features such as authentication logging and so on
- Less flexibility
- Cloud provider may update and break your application

SaaS

- A software running completely in the cloud
- The user doesn't need to install anything on premises or on his machine
  Ex office 365/ salesforce

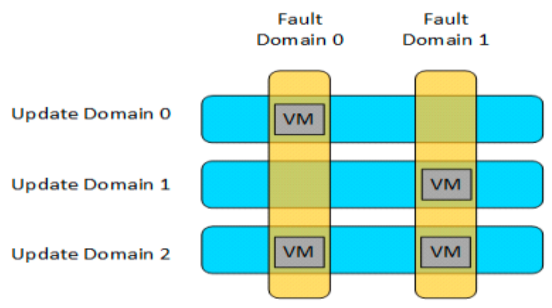
Microsoft Azure is highest growth rate.
The second largest public cloud.

DataCenter: Unique physical building that contains thousands of physical servers that's own power cooling and networking infrastructure.
Azure Geography: An area of the world that contains one or more Azure regions(eg USA,UK, IN)- based on customers/latency/ Compliance(Financial/Health Care/Credit Card for same country)
Azure Region: A region is a set of datacenters deployed within a latency-defined perimeter and connected through a dedicated regional low-latency network. Azure gives you

the flexibility to deploy applications where you need to, including across multiple regions to deliver cross-region resiliency.
Availability zones (AZs) are isolated locations within data center regions from which public cloud services originate and operate. Regions are geographic locations in which public cloud service providers' data centers reside.

Azure Resource: Resources are instances of azure services ex. Virtual Machines, App Services, Storage Accounts etc.

Fault Domains	**Update Domains**
- Each rack of servers is a separate fault domain - VM in the same fault domain share a common power source and physical network switch. - If a power supply or network switch fails, all the servers in that rack fails - Fault domain is a group of resources that may fail at the same time due to the same root cause - **Deploy azure resources in two or more fault domains for high availability.**	- An update domain is a group of resources that can be updated and rebooted if required at the same time - VM in the same update domain will be restarted together during planned maintenance. Azure never restarts more than one update domain at a time. - Some updates require servers to be rebooted - Only one update domain is rebooted at a time.(30 min to recover) - Deploy azure resources in two or more update domains for high availability.



Availability Set:
- Logical grouping for isolating virtual machine resources from each other.
- VMs placed in an availability set run across multiple physical servers compute racks storage units and network switches.

VMSS(Virtual Machine Scale Set):
- VM size

Private and public IP address:

App Services:
- Platform as a service offering
- Use to host webapplication / rest application /backend application
- Without an app service plan you cannot create app service.

## Resource Group:
# Azure CLI and Power Shell
- Every command on azure start with AZ
- Create a resource group --> [az group -l westus clitest-rs-test]
    - Free
    - Development/ test/ prod group
    - Team A resources

Azure provides four levels of management scope: management groups, subscriptions, resource groups, and resources. The following image shows the relationship of these levels.

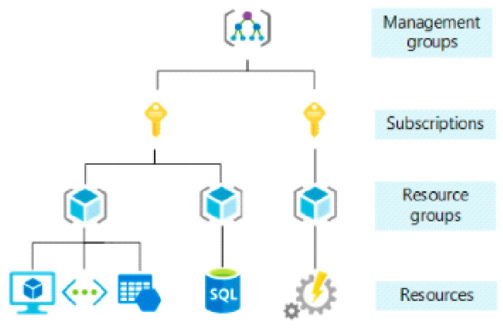


Figure 1: How the four management-scope levels relate to each other.

- **Management groups:** These groups are containers that help you manage access, policy, and compliance for multiple subscriptions. All subscriptions in a management group automatically inherit the conditions applied to the management group.
- **Subscriptions:** A subscription logically associates user accounts and the resources that were created by those user accounts. Each subscription has limits or quotas on the amount of resources you can create and use. Organizations can use subscriptions to manage costs and the resources that are created by users, teams, or projects.
- **Resource groups:** A resource group is a logical container into which Azure resources like web apps, databases, and storage accounts are deployed and managed.
- **Resources:** Resources are instances of services that you create, like virtual machines, storage, or SQL databases.

## Storage Account
## SLA Service level Agreement -The uptime % of cloud service

Azure Products:

- **Compute**
  - **Logic Apps**
  - **ACI- Azure Container Instance**
  - **App service container**
  - **Virtual Machine**
    - **Cost of VM**
      - VM/Disk/IP/Storage
    - **How to reducing the cost of VM**
      - Auto shutdown (Automatically Shutdown the machine when not needed .)
      - Reserved Instances(Allow upfront payment with substantial discount eg usually offered for 1 or 3 years)
      - Spot Instances (machines that run on unused capacity in Azure/ Can be evicted any moment when needed by azure / offers upto 90% discount)
      - Disk Optimization(right disk for the machine/ Default is Premium SSD- The most expensive option/ Non IO-intensive machines can do with standard SSD).
    - **Availability of a VM.**
    - **Virtual Machine Scale set**
      - A group of separate VMs sharing the same image
      - Managed as a group
      - Great for handling unpredictable load
    - Scale set architecture
      - Load Balancer==> Virtual Machine scale sets ==>

  - **Resource Group**
  - **App Services**
      - A fully managed web hosting for websites(HTTP based services)
      - Publish your code- and it juts runs
        - Supports different programming language
      - No access to the underlying servers
      - Secured and compliant
      - Integrates with many source control and devOps engines
    - App service auto scaling.
      - Scall up==> Handling higher loads with one instances
      - Scaling Out ==> Increasing the number of instance
    - Azure DevOps
    - --
    - **Creating App service using command line (from cloud shell)**

----- --

1) Create Web api
   a) Create App service
      i) PS /home/sudarshan> az appservice plan create --name dev-cakes-sn-api --resource-group Learning-az --sku FREE
   b) Create web app
      i) Az webapp create --resource-group Learning-az --plan dev-cakes-sn-api --name cakes-for-all-webapp
         1. https://None@cakes-for-all-sn.scm.azurewebsites.net/cakes-for-all-sn.git

Azure Storage

- Blob storage vs file storage
  - **Blob Storage**
    - Ideal for video audio images and documents
    - Available globally via HTTPS
    - Can Handle large backups and archives
    - Is cheaper than Azure file storage
  - **File Storage**
    - Azure Files offer fully managed File shares in the cloud that are accessible via the industry-standard SMB. Azure File shares can be mounted concurrently by cloud or on-premises deployments of Windows, Linux, and macOS. It can be cached on Windows servers with Azure File Sync for faster access.
- Access Tiers
  - Hot tier - An online tier optimized for storing data that is accessed or modified frequently. The Hot tier has the highest storage costs, but the lowest access costs.
  - Cool tier - An online tier optimized for storing data that is infrequently accessed or modified. Data in the Cool tier should be stored for a minimum of 30 days. The Cool tier has lower storage costs and higher access costs compared to the Hot tier.
  - Archive tier - An offline tier optimized for storing data that is rarely accessed, and that has flexible latency requirements, on the order of hours. Data in the Archive tier should be stored for a minimum of 180 days.

Hot tier	Cool tier	Archive tier	
Availability	99.9%	99%	Offline
Availability (RA-GRS reads)	99.99%	99.9%	Offline
Usage charges	Higher storage costs, but lower access and transaction costs	Lower storage costs, but higher access and transaction costs	Lowest storage costs, but highest access, and transaction costs
Minimum recommended data retention period	N/A	30 days[1]	180 days
Latency (Time to first byte)	Milliseconds	Milliseconds	Hours[2]
Supported redundancy configurations	All	All	LRS, GRS, and RA-GRS[3] only

- **Cosmos DB**
  - Available in every region
  - Automatically Index the data
  - Supports find tunning
  - Scales storage and throughput separately

AKS(Azure Kubernetes Services)

- Allows deploying containers and managing them using kubernetes on Azure
- Paying only on the instances(=VMs) used
  - **Containers:** Thin package model/ Packages software, its dependencies and configuration files/ can be copied between machines
    - Why ? Predictability/ performance/
    - Not containers? ==> Isolations(Containers shares the same OS so isolation is lighter then VM)

  - **Dockers:** The most popular container environments
      - Dockerfile ==> contains instructions for building custom images
  - Kubernetes
    - Provides all aspects of management (container)
    - Kubernetes create command
      - PS /home/sudarshan> az aks create --resource-group Learning-az --name aks-east-sn-test-kubernetes --node-count 1 --generate-ssh-keys
      - To verify the nodes commands
        - Kubectl get nodes

===

Securing your application

- **Role Base vs Claims- Base**
- **Key Vault**
  - Centralized place to store the key password

Azure Networking

==> Never leave a VM open to the internet(Brute force attack on port 3389 RDP/ No line of defense in front of the VM web server)

Virtual Network

-> A network in which you can deploy cloud resources
-> many cloud resources you can deploy within Vnets
ex VM , app services, DBs.
-> Resources in VNet can communicate with each other by default
-> In AWS it is called VPC
-> Each Vnet has its own address range

- **CIDR notation**
- **Sub Net**
  --> Logical segment in the VNets
  --> Share a subset of the Vnet IP range
  --> Use logic group or resources in VNets
  --> VM of different vNet cannot connect

- **Network Security Group(NSG)**
  -->A gatekeeper of subnets
  --> Defines who can connect in and out to subnet
  --> Think of it as a Mini-firewall
  --> Looks at 5 tuples?
  1. Source
  2. Source Port
  3. Destination
  4. Destination Port
  5. Protocol(TCP, UDP both)

  --> This is called security group

- **Setting UP NSG**
- **Peering NSG**
- **Secure VM Access**
  - The large the attack surface- greater the risk
  - Leaving public Ips open is always a risk we need to avoid
  - Not directly related to the app design but important nonetheless
  - **What can be done in AZure?**
    - **JIT Access(Just in time access)**
      - Opens the port for access on demand and automatically closes it
      - Rest of the time - its closed
      - Can be configured from the VM's page in the portal
      - Requires Security Center License upgrade
    - **VPN**
      - A secure tunnel to the Vnet
      - Can be configured so that no one else can connect to the Vnet
    - **Jump Box**
      - Place another VM in the Vnet
      - Allow access ONLY to this Vnet
      - Only one port is open
    - **Bastion**
      - A web based connection to the VM
      - No open port is required
      - Simple and secure
      - It is expensive
      - Requires portal access

- **Service Endpoint**
  - Service endpoint solves this security risk
  - Creates a route from the VNet to the managed service
  - The traffic never leaves Azure backbone
  - Access from the internet can be block

Load Balancer

- Azure service that distributes load and checks health of VMs
- When a VM is not healthy - no traffic is directed to it
- Can be public or private
- Operates at layer 4 of OSI layer
  - Application
  - Presentation
  - Session
  - Transport **[Layer 4- Knows about IP, port protoal TLS and more, Has no knowledge about the actual content]**
  - Data Link
  - Physical

  ----------------------------------Physical Medium------------------------------
- Basic Type and Standard type

☑ **Data In Azure**

Azure Databases:

- **Database on VM**
  - **Full flexibility**
  - **Full control**
- **Azure SQL**
  - **Elastic Pool**
    - **Allows storing multiple database in single**
    -
    -
  - **Managed Instance:**
    -
- **Cosmos DB**
  - Fully managed NoSql database
  - Amazing performance - <10ms for 99% of operations
  - Globally distributed
  - Multiple APIS: SQL, Mongo, Gremlin, Casendra
  - Hierarchical
- **Azure PostgreSQL**

Serverless Characters

Microsoft will actually let you borrow these unused VMs to run code that can run quickly and return a result, and this concept is referred to as serverless computing. Now, it's important to understand that serverless doesn't mean there isn't a computer involved. It just means you aren't using a computer that's explicitly allocated to you. Instead, you're borrowing one of these unused VMs for a very short period of time.

- **Fully Managed by the cloud provider**
- **Pay only while your code runs**
- **Cope with spikes in demand**
  - **Azure Functions** - Function as a Service(FaaS)- A platform for running functions which are simply your code running in response to an event
    - Azure functions triggers
      - ☐ Timer (Run a function on a schedule)
      - ☐ Message (Listen for messages on a queue)
      - ☐ HTTP Request(Implement Webhooks and APIs)
      - ☐ Many more - Blob creations
      - ☐
  - **How Long Can Azure Functions Run?**

    For any Azure Functions, a single Function execution has a **maximum of 5 minutes by default** to execute. If the Function is running longer than the maximum timeout, then the Azure Functions runtime can end the process at any point after the maximum timeout has been reached.
    - **Why Serverless?**
      - ☐ Event Driven stateless programming
      - ☐ Binding and integrations enable rapid development
      - ☐ Very low cost - great for startup prototypes
      - ☐ Rapid and automatic scale out
    - **Disadvantage**
      - ☐ Cold start -
      - ☐ Limited coverage of integration

Azure Function

- **Function App**
  - A function app provides an execution context in which our block of code…function will run. Given this, a function can be thought of as a unit of deployment and management for our functions. It is worth noting that a function app can have more than one function that is managed and scaled together.
  - Also, all functions in a function app must be written in the same programming language and share the same pricing plan and deployment approach.
- **A few trigger examples include**

- HTTP trigger.	A function that will be run whenever it receives an HTTP request, responds based on the payload or query string. In our example, we will be taking a look at this;
Azure Blob Storage trigger Timer trigger. SendGrid. Azure Cosmos DB trigger.	- Azure Blob Storage trigger. A function that will be executed whenever a blob is added to a specified container; - Timer trigger. A function that will be executed on a given schedule; - SendGrid. A function that will send a confirmation email when new items are added to a particular queue; - Azure Cosmos DB trigger. A function that will be executed whenever documents change in a document collection.

function.json file

Tools

- **Azure CLI**

Azure functions

- Small focus functions running as a result of an event

- Great for event driven system
- Automatically managed by the azure
  - Start/ stop/ autoscale
- Flexible pricing plan
- **Serverless**
  - You don't need to think about (VMs\CPU\Memory etc)
- **Cold Start**:
  - Azure function are completely managed by Azure
  - After some time of inactivity azure might take down the function host
  - The next activation function will take time (2-3 second before the cold run)
- Code + configuration file (function.json)
- Once You create a function you deploy them to use via a **function APP**.
- **Pricing model**
  - Consumption Plan
    - Based on Compute usage
    - Only pay when function are running
    - Automatically scales
  - App service Plan
    - Need to host custom image
    - Already have VMs running other app services
- Durable function

- **Azure Function Hosting Plan:**
  - **Consumption Plan**
    - Pay only for what you actually use(Execution time- 0.000016/GB-s /// total execution $0.20 per million executions)
    - Downside(1.5GB RAM , Cold start)
  - **Premium plan**
    - Pay for per warmed instances(hosts)
    - Pay for scale out instances
    - No cold starts
    - No memory limits
    - Better performance
    - Vnet integration
    - Predictable price
    - Downside- more expensive
  - **Dedicated Plan**
    - The function run on an existing app service
    - Great if server is under-utilized
    - No additional cost
    - Downsides- No Auto scale support

- **Durable Functions**

- **Logic Apps- It takes graphical programming**
  - Drag and drop serverless workflows
  - Uses connectors triggers and actions
  - Workflows built using JSON files

- **Container Instances**

ARM Template(Deploy a custom)
- Azure Resource Manager Template
- A json file describing the resource to be created
- Deploy and configure Azure resources
- Declarative
- Specifying dependencies

What's required for on-permises applications?
- Physical computer hardware
  - Expenses of upgrading hardware
  - Repair expenses
- Network Infractructure
- Storage areas for server racks and so on

Benefits of the cloud model
- Rent computing resources from cloud provider
- Shifts expenses from capital expenses to operating expenses.
- Reduced costs- Better economies of scale
- Use only what you need- a consumption based model

Azure Instance Metadata Services:

Download azure Icons

Management Group:

ARM(Azure Resource Manager) templates:

Azure VN Gateway

- **Azure PNet peering**
    - **Traffic between Virtual Network Peering same azure region**
    - **Load balancer internally**

- **List of azure services for basic load balancer**
    - Redis Cache
    - Application Gateway V1
    - Service Fabric
    - Azure SQL Database Managed Instance
    - Active Directory Domain Service
    - Logic Apps
    - HDInsight
    - Azure Batch
    - App Service Environment

<u>Migrating to Microservices:</u>

- **Data Ownership**
- **Tight coupling**
- **Technical Debt**
- **Incremental migration**
- **Communicate vision**

<u>Describe Container(Blob) storage</u>

- Store some graphics file/Video storage
- Block blobs
- Append blobs
- Page blobs
- Store in containers
- Moving data to blob storage

<u>Azure Disk Storage</u>

- Standard Harddrive
- SSD
- Managed and unmanaged

<u>Azure Files</u>

- Disk space in the cloud
- Managed SMB solution
- Azure file sync for faster access

<u>Azure Storage Tiers</u>

1. Hot -Lowest cost for accessing and highest storage
2. Cool - Longer time , lowest storage cost then hot , but higher for accessing cost (at least 30 days storage)
3. Archive - access cost highest (180 days)

<u>Azure Cosmos DB</u>

- NoSQL database system
- Four types of NoSQL database system
    - Key Value System
        - Data tied to unique key
    - Column System
    - Graphical Database- relationship between each piece
    - Document Database - A document database stores data in JSON, BSON , or XML documents

<u>Azure Bot Service:</u>

<u>Monitor :</u>

<u>Application Insight:</u>

Optimizing Your Application:

- Using Redis for caching
- Content caching pattern
    - Eg menu and footer for website
- User session caching
    - Storing user information in cookies
- Using azure content delivery Network
    - Dynamic site acceleration
        - Helps deliver uncacheable content more quickly
        - Route optimization
        - Adaptive image compression
            - Base on the current network image resolution changes
    - HTTPS cdn
    - GIO filtering request
- Kubernetes service scaling strategies
    - Scale pods or nodes
    - Horizontal pod scaler

===

Microservices

Wednesday, September 15, 2021 12:52 PM

★ **What Are Microservices?**
Microservices are independently deployable services modeled around a business domain. They communicate with each other via networks, and as an architecture choice offer many options for solving the problems you may face. It follows that a microservice architecture is based on multiple collaborating microservices.

- Microservices also have the advantage of being technology agnostic.
- Microservices communicate with each other via these networks—making them a form of distributed system.

- **Independent Deployability**
  - we need to be able to change one service without having to change anything else
- **Own Their Own Data**

TIP - Don't share databases, unless you really have to. And even then do everything you can to avoid it.

It's much more important that you focus on two key things. First, how many microservices can you handle? As you have more services, the complexity of your system will increase, and you'll have to learn new skills (and perhaps adopt new technology) to cope with this. It's for this reason I am a strong advocate for incremental migration to a microservice architecture. Second, how do you define microservice boundaries to get the most out of them, without everything becoming a horribly coupled mess?

Monolithic :

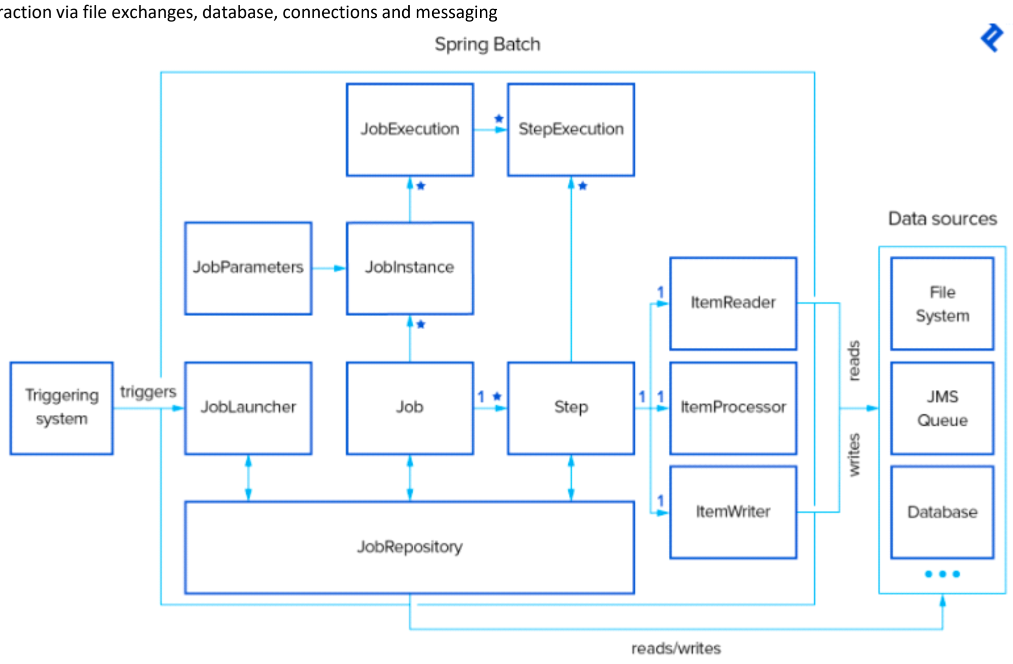
I am primarily referring to a unit of deployment. When all functionality in a system had to be deployed together, we consider it a monolith. There are at least three types of monolithic systems that fit the bill: the single-process system, the distributed monolith, and third-party black-box systems.

Spring Batch

Sunday, September 19, 2021 1:33 PM

What is Batch processing?

- Behind the scenes processes
- Runs without user interaction
- Executes over a fixed data set
- Scheduled jobs correspond with business activities
- Interaction via file exchanges, database, connections and messaging



Spring Batch:

Spring Batch is a lightweight, comprehensive framework designed to facilitate the development of robust batch applications.

a batch process is typically encapsulated by a Job consisting of multiple Steps. Each Step typically has a single ItemReader, ItemProcessor, and ItemWriter. A Job is executed by a JobLauncher, and metadata about configured and executed jobs is stored in a JobRepository.Each Job may be associated with multiple JobInstances, each of which is defined uniquely by its particular JobParameters that are used to start a batch job. Each run of a JobInstance is referred to as a JobExecution.
Each JobExecution typically tracks what happened during a run, such as current and exit statuses, start and end times, etc. A Step is an independent, specific phase of a batch Job, such that every Job is composed of one or more Steps. Similar to a Job, a Step has an individual StepExecution that represents a single attempt to execute a Step. StepExecution stores the information about current and exit statuses, start and end times, and so on, as well as references to its corresponding Step and JobExecution instances.

An ExecutionContext is a set of key-value pairs containing information that is scoped to either StepExecution or JobExecution. Spring Batch persists the ExecutionContext, which helps in cases where you want to restart a batch run (e.g., when a fatal error has occurred, etc.). All that is needed is to put any object to be shared between steps into the context and the framework will take care of the rest. After restart, the values from the prior ExecutionContext are restored from the database and applied.
JobRepository is the mechanism in Spring Batch that makes all this persistence possible. It provides CRUD operations for JobLauncher, Job, and Step instantiations. Once a Job is launched, a JobExecution is obtained from the repository and, during the course of execution, StepExecution and JobExecution instances are persisted to the repository.

Azure steps AZ-900 exam guide

Wednesday, September 22, 2021 9:55 AM

- Azure-900
  - Exam Guide - content Outline
    - Cloud Concepts-- 25-30%
    - Azure architecture and services - 35-40%
    - Azure Management and governance - 30-35%
    - Security privacy Compliance and trust - 10-15%
    - Pricing and support -- 20-25%
  - =================================
    - Passing grade is 700/1000- 75%
    - There are 40-60 Questions
    - Duration of 60 mins
    - Valid for 24 months
  - What is Cloud computing?
    - Cloud computing is the delivery of computing services—including servers, storage, databases, networking, software, analytics, and intelligence—over the Internet ("the cloud") to offer faster innovation, flexible resources, and economies of scale. You typically pay only for cloud services you use, helping you lower your operating costs, run your infrastructure more efficiently, and scale as your business needs change.
    - Public Cloud
      - Owned and operated by a third party cloud service providers eg - azure
    - Private cloud
      - A private cloud refers to cloud computing resources used exclusively by a single business or organization.
      - Private cloud can be physically located on the company's on-site datacenter.
    - Hybrid Cloud
      - Combination of both private and public clouds. Allows data and applications to be shared between them.

On-Premise	Cloud Providers
- You own the servers - You hire the IT people - You pay or rent the real-estate - You take all the risk	- Someone else owns the servers - Someone else hires the IT people - Someone else pays or rents the real-estate - You are responsible for your configuring cloud services and code, someone else takes care of the rest

==15.10 sec

  - **Advantages of cloud**
    - ❑ Cost
    - ❑ Agility
    - ❑ Service Quality
    - ❑ Integration of latest technology – IOT & ML
    - ❑ High Availability
    - ❑ Reliability with Real time failover
    - ❑ Disaster recovery
    - ❑ Ease of Management
  - CapEx vs OpEx
    - CapEx(Capital Expense) - It is a spending of money on **physical infrastructure up front** to create a benefit in the long term. Example - Server costs, Storage costs, Network costs, backup and archive costs.
    - Operating Expenses(OpEx) - It is an expense **required for the day to day functioning** of a business. OpEx is spending money on services or products now and being billed for them now. There's no upfront cost. Ex lease/rent storage in data center, Leasing software

For test- before exam--
[2020.7.27]AZ-900.pdf - Google Drive

AZ-900: Microsoft Azure Fundamentals Sample Questions | Microsoft Docs

Exam AZ-900: Microsoft Azure Fundamentals – Skills Measured

Download eclipse as a zip
Steps:

1. In Eclipse, select the File menu, then select New -> Maven Project.
2. Accept the defaults in the New Maven Project dialogue and select Next.
3. Find and select the azure-functions-archetype and click Next.
4. Be sure to fill in values for all of the fields including resourceGroup, appName, and appRegion (please use a different appName other than fabrikam-function-20170920120101928), and eventually Finish.

Run functions locally in the IDE

Note
Azure Functions Core Tools, version 2 must be installed to run and debug functions locally.

1. Right-click on the generated project, then choose Run As and Maven build.
2. In the Edit Configuration dialog, Enter package in the Goals, then select Run. This will build and package the function code.
3. Once the build is complete, create another Run configuration as above, using
   a. azure-functions:run as the goal and name. Select Run to run the function in the IDE.

azure-functions:run	-e
azure-functions:deploy	
azure-functions:run -DenableDebug	

```
C:\Program Files (x86)\Microsoft SDKs\Azure\Storage Emulator>cmd /K AzureStorageEmulator.exe help
Windows Azure Storage Emulator 5.10.0.0 command line tool
Usage:
  AzureStorageEmulator.exe init          : Initialize the emulator database and configuration.
  AzureStorageEmulator.exe start         : Start the emulator.
  AzureStorageEmulator.exe stop          : Stop the emulator.
  AzureStorageEmulator.exe status        : Get current emulator status.
  AzureStorageEmulator.exe clear         : Delete all data in the emulator.
  AzureStorageEmulator.exe help [command]  : Show general or command-specific help.

See the following URL for more command line help: http://go.microsoft.com/fwlink/?LinkId=392235

C:\Program Files (x86)\Microsoft SDKs\Azure\Storage Emulator>
```

Issues fixed on
-- issue on local.setting.xml

```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "UseDevelopmentStorage=true",
    "FUNCTIONS_WORKER_RUNTIME": "java"
  },
}
```

1. Build clean
2. Build package
3. Build --> azure-function:run

-=-----------------
Storage Emulator
0 */1 * * * * ==> run every minute

Codings

Wednesday, November 10, 2021 6:09 AM

Title	Codes

Azure Blog Storage	public static void main(String[] args) throws IOException {
	String connectStr = "DefaultEndpointsProtocol=https;AccountName=aasalesbxdevstorage;AccountKey=igTv3Z7tqJm6GiAvpdx2LiEhIuBoHZUlB1u2xA0+Yswo64OBqud46Ed5NMEh5y/oMVcSYIpRfb+JJBvbNQn0iw==;EndpointSuffix=core.windows.net";
	// Create a BlobServiceClient object which will be used to create a container
	// client
	BlobServiceClient blobServiceClient = new BlobServiceClientBuilder().connectionString(connectStr).buildClient();
	
	String containerName = "bxjobs";
	
	// Create the container and return a container client object
	BlobContainerClient containerClient = blobServiceClient.getBlobContainerClient(containerName);
	System.out.println("New container created");
	
	String localPath = "./data/";
	String pattern = "yyyyMMdd";
	SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
	
	String date = simpleDateFormat.format(new Date());
	String fileName = "BX_COOLEFFECT_REDEEM_RPT_" + date + ".csv";
	// File localFile = new File(localPath + fileName);
	
	// Write text to the file
	FileWriter writer = new FileWriter(localPath + fileName, true);
	writer.write("Hello, World!");
	writer.close();
	
	// Get a reference to a blob
	BlobClient blobClient = containerClient.getBlobClient(fileName);
	
	System.out.println("\nUploading to Blob storage as blob:\n\t" + blobClient.getBlobUrl());
	
	// Upload the blob
	blobClient.uploadFromFile(localPath + fileName);
	System.out.println("File upldaed successfully !!!");
	
	for (BlobItem blobItem : containerClient.listBlobs()) {
	    System.out.println("File list with name ==>");
	    System.out.println("\t" + blobItem.getName());
	}
	}

Containers	Test

Devops

Monday, January 17, 2022 2:19 PM

Git Hub Action:

- **Working on YAML**
- **Workflow and Action Attributes**
  - **Name**
    - **The name of the workflow**
    - **Not Required**
  - **On**
    - **The Github event that triggers the workflow**
    - **Required**
      - **Repository Events**
        - **Push**
        - **Pull_request**
        - **Releases**
      - **Webhooks**
        - **Branch creation**
        - **Issues**
        - **Members**
      - **Scheduled**
        - **Cron format**
  - **Jobs**
    - **Workflow must have at lest one job**
    - **Each job must have identifier**
    - **Must start with a leetter or underscore**
  - **Runs-on**
    - **The type of machine needed to run the job**
  - **Steps**
    - **Lets of action or commands**
    - **Access to the file system**
    - **Each step run s in its own process**
  - **Uses**
    - **Identifies an action to use**
    - **Defines the location of that action**
  - **Run**
    - **Runs commands in the virtual environment shell**
  - **Name**
    - **An optional identifier for the step**

```
name: Api CI/CD
on:
  push:
    branches:
      - master
  workflow_dispatch:
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: just echo a message
      run: |
        echo Hello there!
deploy_to_dev:
    runs-on: ubuntu-latest
    needs: build
    if: github.event_name != 'pull_request'
    environment: development
    steps:
    - name: just echo a message for deploy to dev
      run: |
        echo Hello there!
deploy_to_production:
    runs-on: ubuntu-latest
    needs: deploy_to_dev
    if: github.event_name != 'pull_request'
    environment: prod
    steps:
    - name: just echo a message
      run: |
        echo Hello there!
```

- **Adding dependencies(needs)-** Identifies one or more jobs that must complete successfully before a job will run
  - **Jobs (need attribute)- job1 depends on job2**
    - **Job1**
    - **Job2**
    - If **job3 needs** job1 and job2
      - Job3
        - needs: [job1, job2]
  - https://github.com/Sudarshan-Neupane/SpringAngular/edit/master/.github/workflows/maven-publish.yml
- Environment variables
  - Are case sensitive
  - **Environment Variables can be defined on**
    - Workflows
    - Jobs
    - And steps level
      - Shell variable syntax
        - Bash(linux/macOS)
          - $VARIABLE_NAME
        - PowerShell(Windows)
          - $Env:VARIABLE_NAME
        - YAML syntax
          - ${{env.VARIABLE_NAME}}

  - Storing secrets
    - Stored as encrypted environment variables
    - Cant be viewed or edited
    - Workflows can have up to 100 secrets
    - Secrets limited to 64KB
      - ${{ secrets.DISPLAY_NAME}}
- Artifacts
  - Pass data between workflow jobs
    - Job1 - created and upload artifact
    - Job2
      - wait for job 1 to complete
      - Download and use artifacts
        - Job1:
          - ....

- ………

Job1	Job2	Job3
…………….. …………….. Name: Upload artifact for linux Uses: actions/upload-artifact@v1.0.0	Job2 Wait for job1 needs to complete - Name: Download artifacts - Uses: actions/download-artifact@v1.0.0	Wait for job1 needs to complete - Name: Download artifacts - Uses: actions/download-artifact@v1.0.0

- **Pull Request(PR management with github actions)**
  - Automatically approve and merge PRs based on Criteria
  - Run automated tests to check the code in the PR
  - Check the username that submitted the PR
  - Approve and merge the PR
  - Delete the branch associated with PR
    - On:
      - Pull_request:
        - Types: [opened, reopened]

- **Continuous Integration :**
  - **Linting(coding standard) and unit test**
  - **Build(compile code into a binary package)**

- **Docker File Review**

Docker

Docker is a software framework for building, running, and managing containers on servers and the cloud. Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment
Docker makes it really easy to install and run software without worrying about setup or dependencies.
Containers simplify delivery of distributed applications and have become increasingly popular

Normal Flow for installing software
==> download Installer ==> run Installer ==> Get an error message during installation ==>(a) troubleshoot issue ==> Rerun Installer ==>get another error go to (a)

1. Install docker on windows

- First create a dockerFile

  Create a build file on docker
  - Docker build .

  It will create the image with the random image id.
  Now tag the image:
  - docker build -t <dockerId>/<nameOfTheProject> .
  - Start Image
    - docker run <dockerId>/<nameOfTheProject>
    - If you start the project with specific port and map the docker port to your application port -
      - Port mapping
        - Docker run -p 8080:8080 <imageId>
      - Specifying the working directory in Docker - list all the files inside the docker
        - docker -it <imageId> sh
          - ls

  Creating and running a container from a image
  - Docker run <image name>
  - Docker run busybox ls
  - **docker run <image name> command!!!**

  i. List all running containers
     1) Docker ps
  ii. Docker running container
     1) Docker run -it imageid
  iii. Docker create new Image
     1) Dockerfile --> docker Client --> Docker Server --> Usable Image
  iv. Docker build only the change single file

- **Docker Compose ------ compose for more than one container -- two container does not have automatic connection**
  - **-- docker run redish**
  - **-- second terminal docker run <image-name>**
  - Create a separate cli file called docker-compose.yml
    - Steps

- Here are the containers I want to create
  - Radis-server
    - Make it using the radis image
  - Node-app
    - Make it using the dockerfile in the current directory
    - Map port 8080 to 8080

- Docker run background
  - Docker run -d radis
  - Docker ps
  - Docker stop dockerId
  - Docker-compose down --- to stop all running container

- How to deal with the container if it is crashed (Maintenance)

==

<u>VM Vs Container</u>

<u>Kubernetes: ()</u>

<u>Pod:</u> Pods are the smallest unit of compute

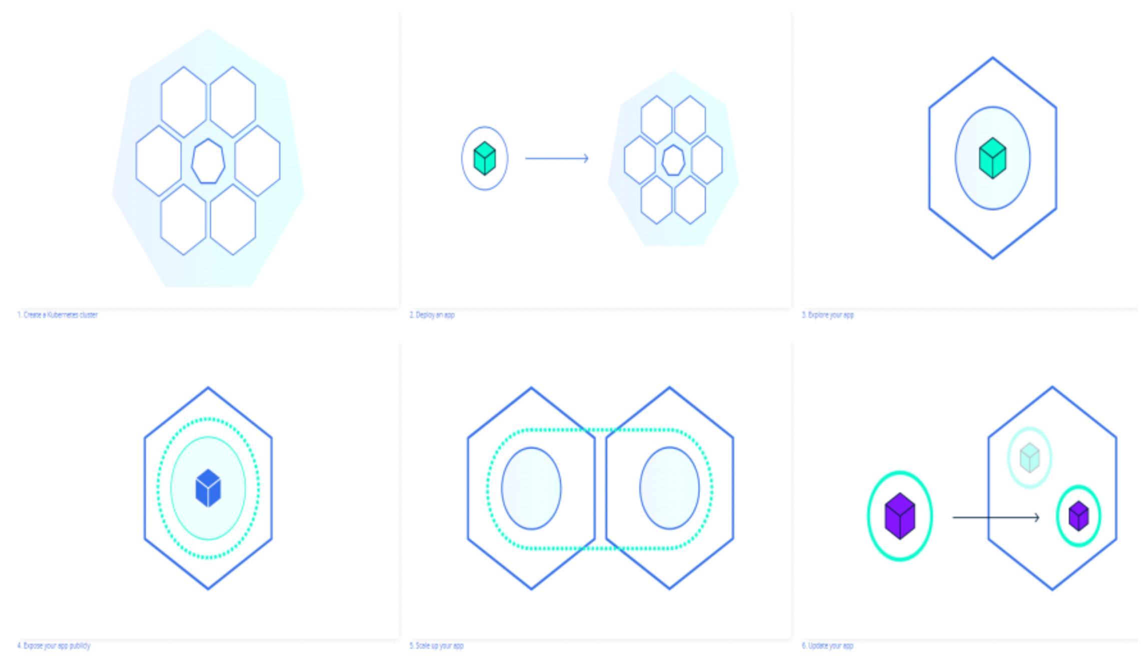
Kubernetes orally ==> github.com/sandervanvugt/handsonkubernetes

What can Kubernetes do for you?

With modern web services, users expect applications to be available 24/7, and developers expect to deploy new versions of tho se applications several times a day. Containerization helps package software to serve these goals, enabling applications to be released and updated w ithout downtime. Kubernetes helps you make sure those containerized applications run where and when you want, and helps them find the resource s and tools they need to work. Kubernetes is a production-ready, open source platform designed with Google's accumulated experience in container orchestration, combined with best-of-breed ideas from the community.

From <https://kubernetes.io/docs/tutorials/kubernetes-basics/>

Kubernetes Basics Modules



Kubernetes Clusters

Kubernetes coordinates a highly available cluster of computers that are connected to work as a single unit. The abstractions in Kubernetes allow you to deploy containerized applications to a cluster without tying them specifically to individual machines. To make u se of this new model of deployment, applications need to be packaged in a way that decouples them from individual hosts: they need to be containerize d. Containerized applications are more flexible and available than in past deployment models, where applications were installed directly onto specific machines as packages deeply integrated into the host. **Kubernetes automates the distribution and scheduling of application containers across a cluster in a more efficient way.** Kubernetes is an open-source platform and is production-ready.

A Kubernetes cluster consists of two types of resources:

- The **Control Plane** coordinates the cluster
- **Nodes** are the workers that run applications

Cluster up and running

https://kubernetes.io/docs/tutorials/kubernetes-basics/create-cluster/cluster-interactive/
We already installed minikube for you. Check that it is properly installed, by running the *minikube version* command:
`minikube version`
OK, we can see that minikube is in place.
Start the cluster, by running the *minikube start* command:
`minikube start`

- **kubectl get** - list resources
- **kubectl describe** - show detailed information about a resource
- **kubectl logs** - print the logs from a container in a pod
- **kubectl exec** - execute a command on a container in a pod

Rolling updates allow Deployments' update to take place with zero downtime by incrementally updating Pods instances with new ones. The new Pods will be scheduled on Nodes with available resources.

On prem Kubernetes:

Kubernetes Cloud -- runs container

FiberUi
Material ui

Software Engineer Interviews

Sunday, May 8, 2022 2:31 PM

How to prepare for your software engineering interview:

a. Maximize your chances of being shortlisted
b. Find out the interview format
    i. Quiz
    ii. Online Coding Assessment
    iii. Take home assignment
    iv. Onsite
c. Pick a programming language
d. Sharpen your Computer Science fundamentals for interviews
e. Practice for the coding interview ==> Three Month(2-3 hours a day)
f. Prepare for the systems design interview (for mid/senior levels)
g. Prepare for the behavioral interview
h. Negotiating the offer package