Week 4 Study Guide

Key Concepts:

|  |  |
| --- | --- |
| * DevOps | * SDLC |
| * Agile/scrum * CSS * Continuous delivery/deployment * SaaS, PaaS, IaaS * VSTS * Static analysis * Azure AppService * Azure Key Vault | * Continuous integration * Rules * Cloud services * Dev pipeline * Build Server concept * Sonar Qube/Sonar Cloud * Code coverage * Azure VM |

DevOps

Software development lifecycle (SDLC)

Requirements for the project

design plan

build the project (coding)

testing the project

deployment of project. move away from development machines

maintence of the project

Models of SDLC

-Waterfall (goes top to bottom of the SDLC)

Approiate for very mission critical projects

Take a long time in each stage before advancing

Not very responsive to changes

-big bang (do all lifecycle at once)

good for prototyping a project

-iterative (have the ability to go back to stages of lifecycle)

general term for any process where we go back to a previous stage

-spiral (when you get to the end of the lifecycle and go back to the begining)

still doing a stage all at once. seperates it from agile

-agile (breaks project up to little pieces and execute those one at a time)

constantly doing all stages of the lifecycle on different parts of the project

not burdened to design the whole project before going back

Main way we are going to implement agile is with through the scrum method

-Scrum

What is a sprint? A sprint is 2-6 weeks and in DevOps is a plan of those weeks.

At the end you evaluate if you got what you planned done.

Planning and review

-User stories- a piece of functionality that the user should be able to do

- User issues - a bug that needs to be fixed

These 2 build up sprints.

Point values are assigned to a value of expected effort (done in planning)

Scrum cont. (know terms from day 15)

Capacity (in relation to a sprint) - refers to the number of expected productive man hours during the sprint

Velocity - how many points did we really finish this sprint

Daily Standup - On a daily basis, project team will get together for a quick meeting. Tell the group what they accomplished the previous day, what their immediate goal is that day and what is blocking them from progressing

Scrum board (made up of columns)

each column represents a state of the user stories

1. Backlog

2. Sprint

3. In Progress

4. Testing

5. Ready for release

6. Release

github has a scrum board style feature. will be used in project 2

Other types of agile

CMMI

Kanban - board is very important, only show much in each column. Prevents bottleneck. development is continuous.

Project 1 published online http://project1webapp20180717094801.azurewebsites.net/

VSTS (Visual Studio Team Services)

how to take connection string out of appsettings

usually right click project start-up and click manage user secrets

put connection string in there

in team enivornment, everyone will clone repository and create that secret

Project 2

Your choice of application

Tech stack: everything from project 1 + new technologies

Complexity: At least as complex as pizza store with more complex pizza

User accounts with login

Will use service oriented architecture. More than one server

on-premsises deployment (no cloud)

whole culture is called DevOps

physical servers at the company where the application was hosted

Devs <==> Code repo

With DevOps and continuous integration (code repo -> build server -> package repo)

Package repo important (look up) - a repository for full builds, instead of just small updates to source code

Devs -> IT

Devs email compiled code to IT dept

IT manages servers (Dev, Test, Staging, and Production)

Application is updated in big batches

Dev Server:

Published code

Current state of the application

(Operations/Ops)

Test Server:

Focuses on user end tests of application

controlled by QA

Staging Server:

Production Server:

Public internet

DevOps - Extension of agile development passed just the devs, all the way to deployment

Bring Devs and Ops together to agree on those automated processes and the build pipeline

Build server enabling - VSTS is our build server

Continuous integration - constantly inregrating new code from each developer

This is contained within the Devs (code repo and package repo)

We have automation of a couple different things with continuous integration (version control Git, building VSTS, testing xUnit, analysis SonarCloud, packaging VSTS)

The path described previously is called the build pipeline

DevOps Engineers create this pipeline

Notification on failure (important for project 2)

We need to send our lastest version of our code through the pipeline

Continuous Delivery - deploy to servers with human evaluation

Continuous Deployment - deploying fully automatically (no human approval)

Tools that allow continuous integration

Jenkins

TravisCI

VSTS

GoCD

We have now moved on premsises servers to the cloud

Abbreviations CI for continuous integration and CD for Continuous Deployment/Delivery

---------

Cloud Services

Things on the cloud:

DB

Whole Servers (VM)

Storage

Email

Host Webapps

These require abstraction for the end user

Think of cloud services as vendor vs consumer

Vendors: Amazon (AWS), Microsoft (Azure), Google (Google Cloud Platform)

IaaS (Infrastructure as a service)

General category of cloud services, where we have as much access as we can.

We have access to a server and control what is running is on that server, and what is connected to that server, etc.

It has the most control for the consumer. You can control disk, network, memory, all software and OS. Hardware still abstracted

The main example is Virtual Machines (Azure VM, Amazon EC2)

PaaS (Platform as a service)

Somewhere in between SaaS and IaaS. A deployment environment for applications with already provided software, runtimes, etc.

More foolproof, sometimes don't need more control for the user and keep them somewhat abstracted

No access to OS, disk, memory, etc.

Examples (Azure App Service, Amazon Elastic Beanstalk, Google App Engine)

SaaS (Software as a service)

Category of cloud services where the service is software.

The abstract from the consumer is the highest. Used for email and web apps

User doesn't know how or in what language the software was built on

SaaS is software that doesn't run on your computer

Examples (gmail, github)

Sometimes we refer to SQL server as DBaaS (database as a service)

Hybrid cloud - describes where some things are on the cloud but some things will be on premsises.

vendors provide:

- regular cloud services

- government versions of those networks

Multi-Cloud

-services from more than one vendor (DB from amazon, run app on Azure)

Also sometimes running your app on multiple services

-------

Azure storage services

Uses Blob storages - unstructure set of files

-object store (images, audio/video streaming, huge files)

Other parts of Azure:

Azure Files

Azure Disk

Azure Managed Disk

Difference between Files and Disk is that Files can be attached to many machines at once (many VMs), but Disk can only be attached to one machiens at a time

Amazon's version is called S3

As cloud services evolve uptime/downtime requirements have been increased

We manage uptime/downtime with service level agreement(SLA).

This describes the agreement between vendor and consumer of uptime/downtime percentage

Uptime is usually agreed to be around 99.95%. 99.99% less than an hour per year

-------

Regions

-Central US

-West US

-East US

-etc.

Availability zones

-multiple in each region, as independent from each as possible.

-different physical locations

-power

-if one goes down, others probably still up

VSTS example

Slack has apps.

Visual Studio Team Services app. allows team members to get notifications

slack app will tell us url to add to VSTS

Service hooks -> Slack

DevOps cont.

Code coverage in VSTS

Code coverage enabled in VSTS

In every .proj file need

<PropertyGroup>

<DebugType>Full</DebugType>

</PropertyGroup>

Set up quality gate profiles for the project

adjust/remove code coverage to begin

Security Group

A set of rules that describe the IP address that are allowed to connect on specific ports

Benefit is that we can apply it to more than one server

Azure services

Project 2

Tech Stack:

-Everything from Project 1 (Azure SQL Server, TSQL, Entity Framework Core, C#, ASP.NET MVC, xUnit, NLog)

-Deploy to Azure App Service

-VSTS Build pipeline (include Sonarcube/Sonarcloud with code coverage)

-Service Oriented Architecture (break up our solution into 2 services: back end and front end). Put a service layer with a web API

-Angular will be part of front end making requests from back end

-ASP.NET API

Research Angular for Project 2

Need User accounts with authentication, using ASP.NET Identity

DB requirements:

-At least 6 tables

-At least 1 many-to-many relationship

Front end:

-at least 12 views