Tutorial 1

Class	IAS
Date	@Jan 6, 2021
Type	Tutorial

CSE 563

▼ Team Formation

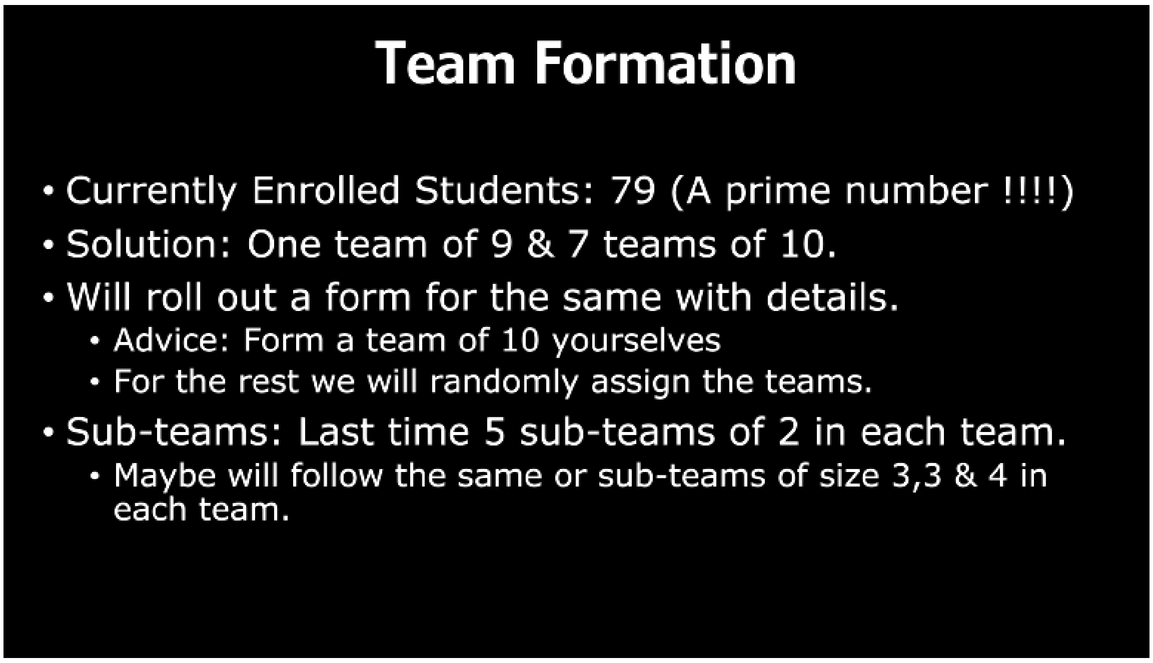
Total students enrolled this time: 79

Team will have around 10 members

Last year each team had 10 members and there were 5 sub teams each of size 2.

Assignments are to be done individually.

Proposal this time: Team of 10 and sub teams of 3.



Team Formation

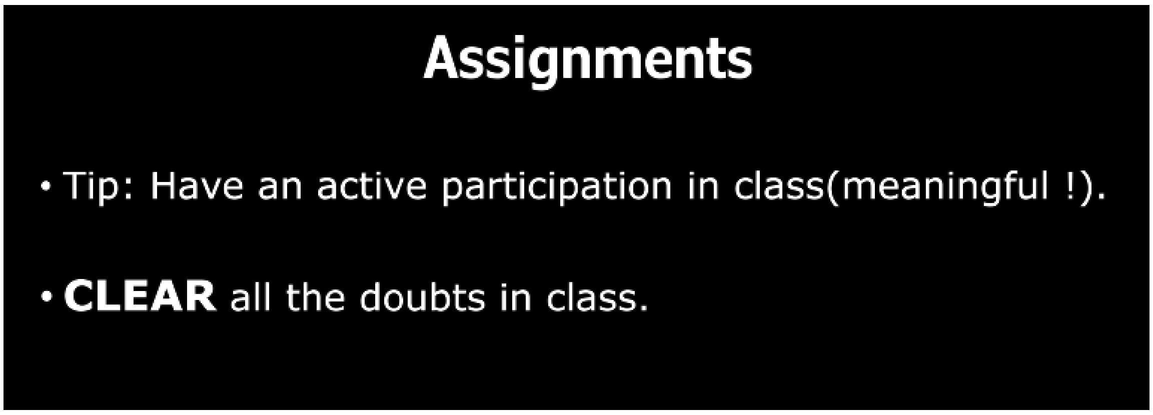
- Currently Enrolled Students: 79 (A prime number !!!!)
- Solution: One team of 9 & 7 teams of 10.
- Will roll out a form for the same with details.
  - Advice: Form a team of 10 yourselves
  - For the rest we will randomly assign the teams.
- Sub-teams: Last time 5 sub-teams of 2 in each team.
  - Maybe will follow the same or sub-teams of size 3,3 & 4 in each team.

▼ Project

Service lifecycle, scheduler, load balancer etc could be different components in the group. Each sub team could work on different components. Every team will work on the same project. How well formed the project is and how are edge cases covered?

Python seems to be a better programming language. Usually, every team works on the same language. Flask, Django, Docker and everything can be explored along with Python.

▼ Assignments

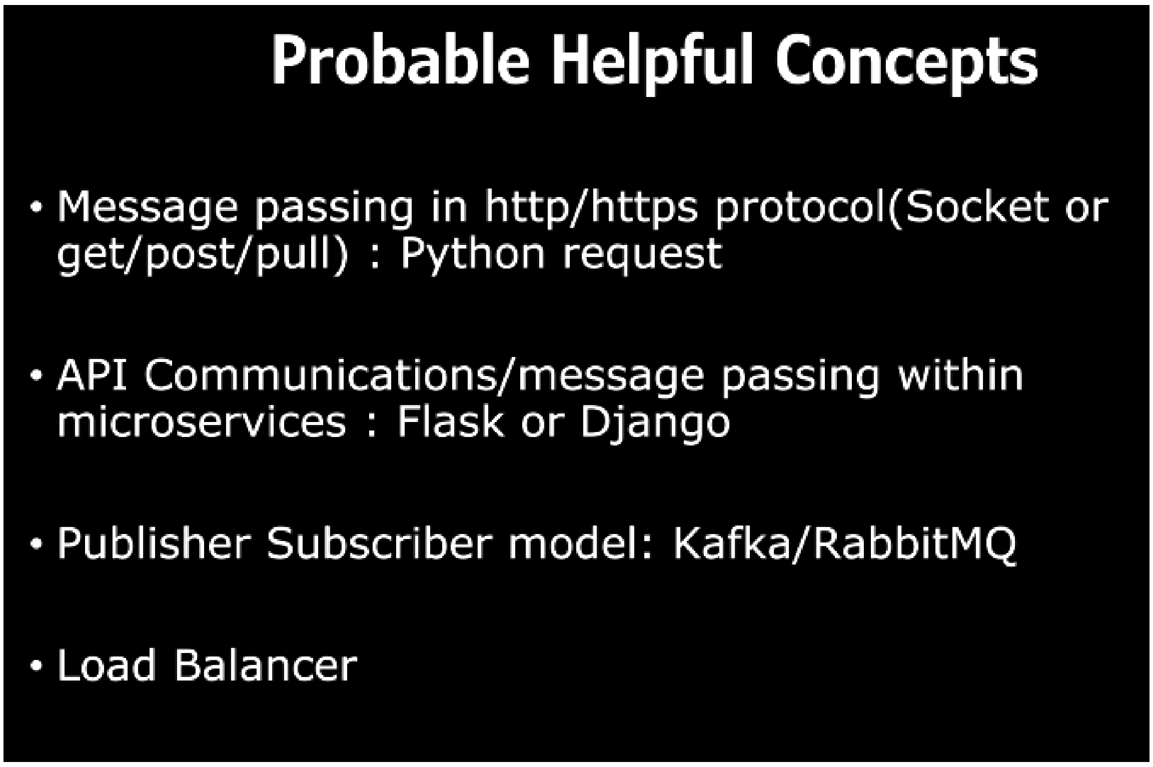


Assignments

- Tip: Have an active participation in class(meaningful !).
- **CLEAR** all the doubts in class.

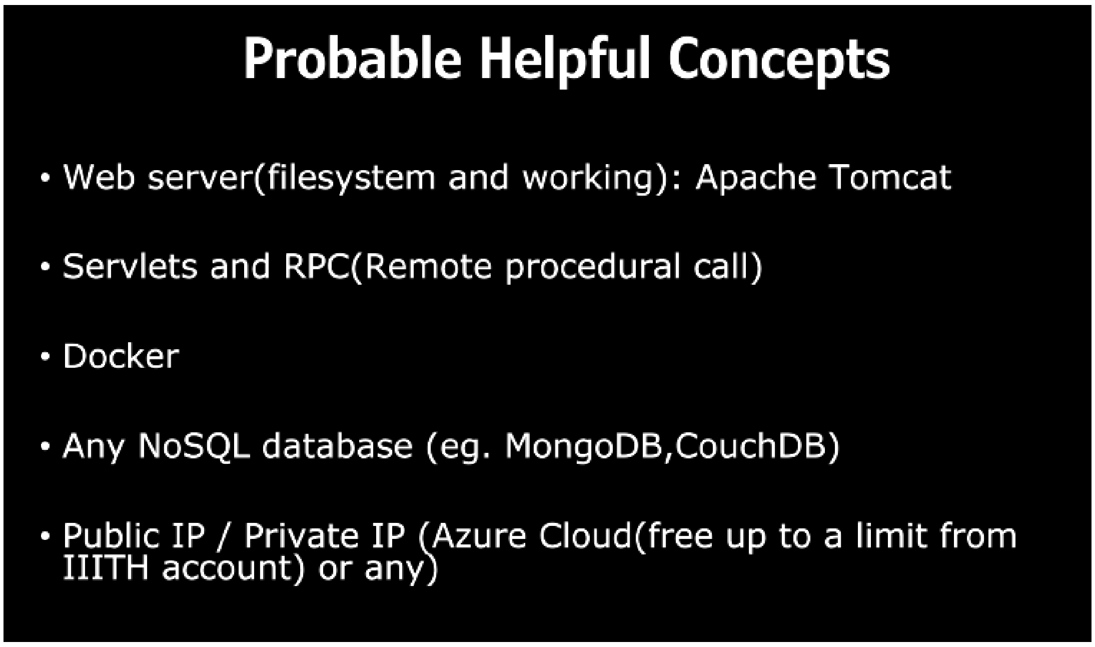
Say,HTTP, middleware etc is going on in the class and if the class is not interactive, Sir might draft an assignment with 2-3 hours deadline.

▼ **Helpful Concepts/Topics**


Probable Helpful Concepts

- Message passing in http/https protocol(Socket or get/post/pull) : Python request
- API Communications/message passing within microservices : Flask or Django
- Publisher Subscriber model: Kafka/RabbitMQ
- Load Balancer

- Python **requests** library
- Sockets can be used but TAs suggest Flask and Django for message passing
- **Kafka**: Works as a data line for communication between different components. Get through producing/consuming of messages (from queue?) in it.
- **Load Balancer:** Build a load balancer from scratch. **haproxy, Nginx** are good platforms and TA prefer using these tools rather than building them from scratch. We will probably be forced to create a load balancer from scratch. But these tools will give us an idea how a load balancer should be.


Probable Helpful Concepts

- Web server(filesystem and working): Apache Tomcat
- Servlets and RPC(Remote procedural call)
- Docker
- Any NoSQL database (eg. MongoDB,CouchDB)
- Public IP / Private IP (Azure Cloud(free up to a limit from IIITH account) or any)

Look into how Apache Tomcat works. We will not be able to build an assignment on it in 3-4 hours. Have an idea beforehand.

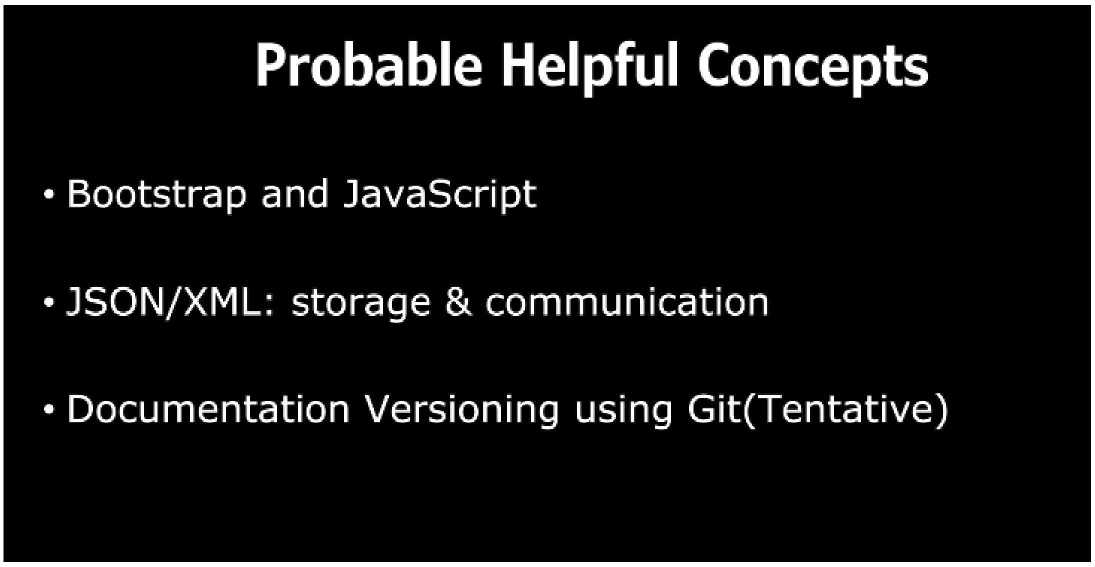
Assignments on **Servlets and RPC, Tomcat** would mostly be there. They are there almost every year. How runtime file is created in Servlets.

Docker: Docker Daemon, Docker container, Images would be enough.

NoSQL DB: MongoDB or any other NoSQL DB.

Also look into **synchronous and asynchronous** programming

When to use SQL vs NoSQL.



Probable Helpful Concepts

- Bootstrap and JavaScript
- JSON/XML: storage & communication
- Documentation Versioning using Git(Tentative)

Having a good UI(bootstrap, JS) will be helpful in explaining the project to Sir in the end.

JSON/XML:

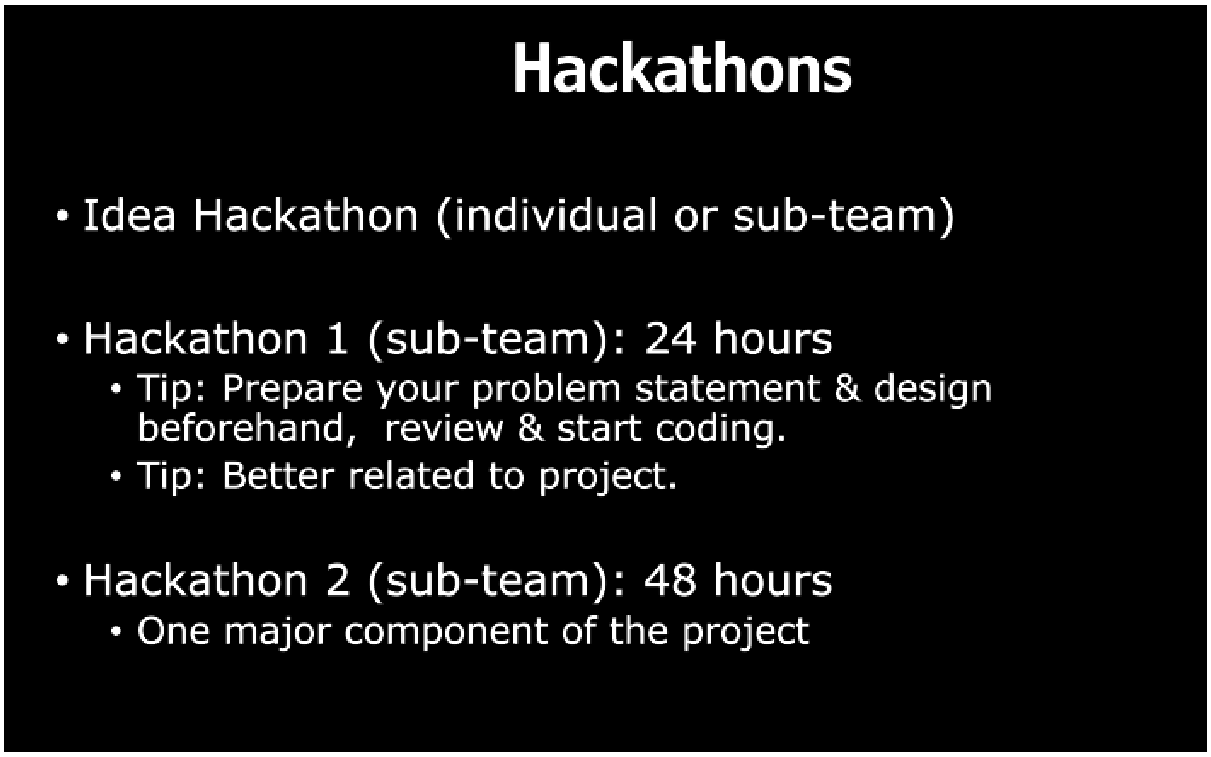
▼ How to communicate b/w two machines which are on diff private networks. Communication through sockets and handshaking will work only in the same network. We can create a separate node(public) on the cloud and the machines in different n/w can communicate through that cloud node. Azure has a free usage for a month or so. Activate it around March beginning.

▼ **Hackathons**

Our idea should be finalized before hackathon 1. We will try to build something related to project in hackathon 1. It might not be covering some scenarios.

Hackathon 2 will be a 2 day event. Let's say the whole team has decided to come up with 10 components. Say subteam 1 would work on 2 of those components and they might not cover all the cases but still.

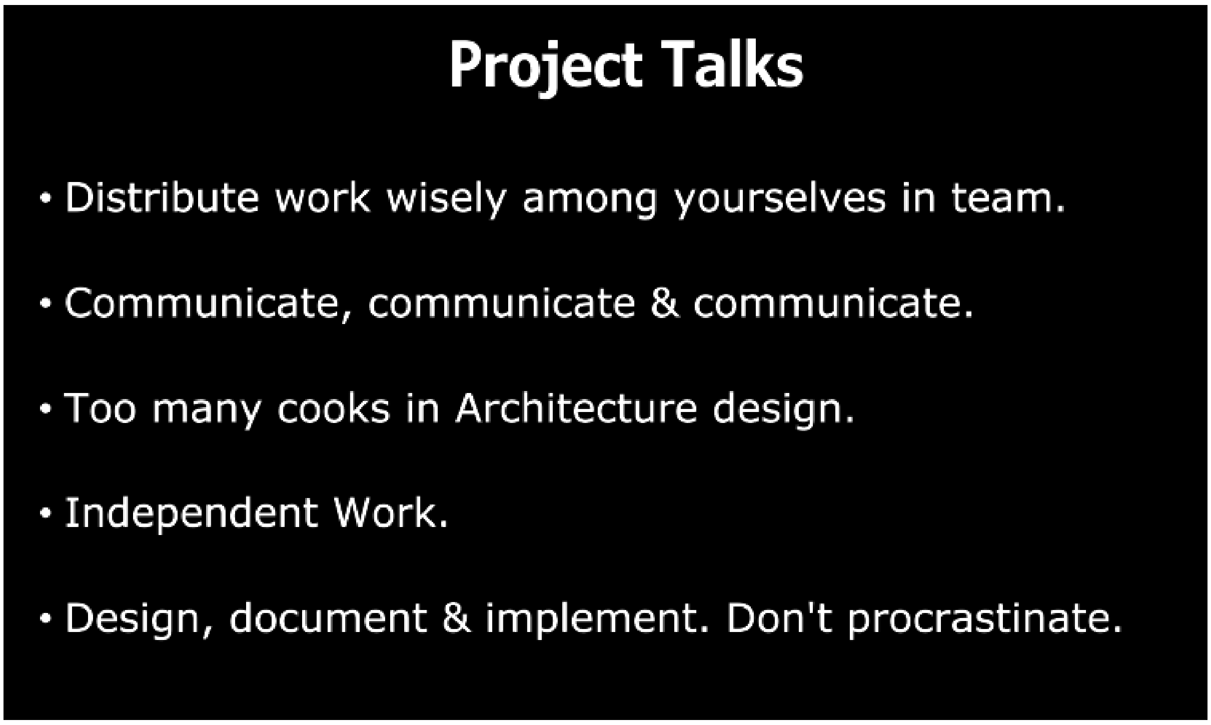
▼ Setup ssh on linux and/or putty on windows for accessing remote machines.



Hackathons

- Idea Hackathon (individual or sub-team)
- Hackathon 1 (sub-team): 24 hours
  - Tip: Prepare your problem statement & design beforehand, review & start coding.
  - Tip: Better related to project.
- Hackathon 2 (sub-team): 48 hours
  - One major component of the project

2nd hackathon will be around March end/ April start.

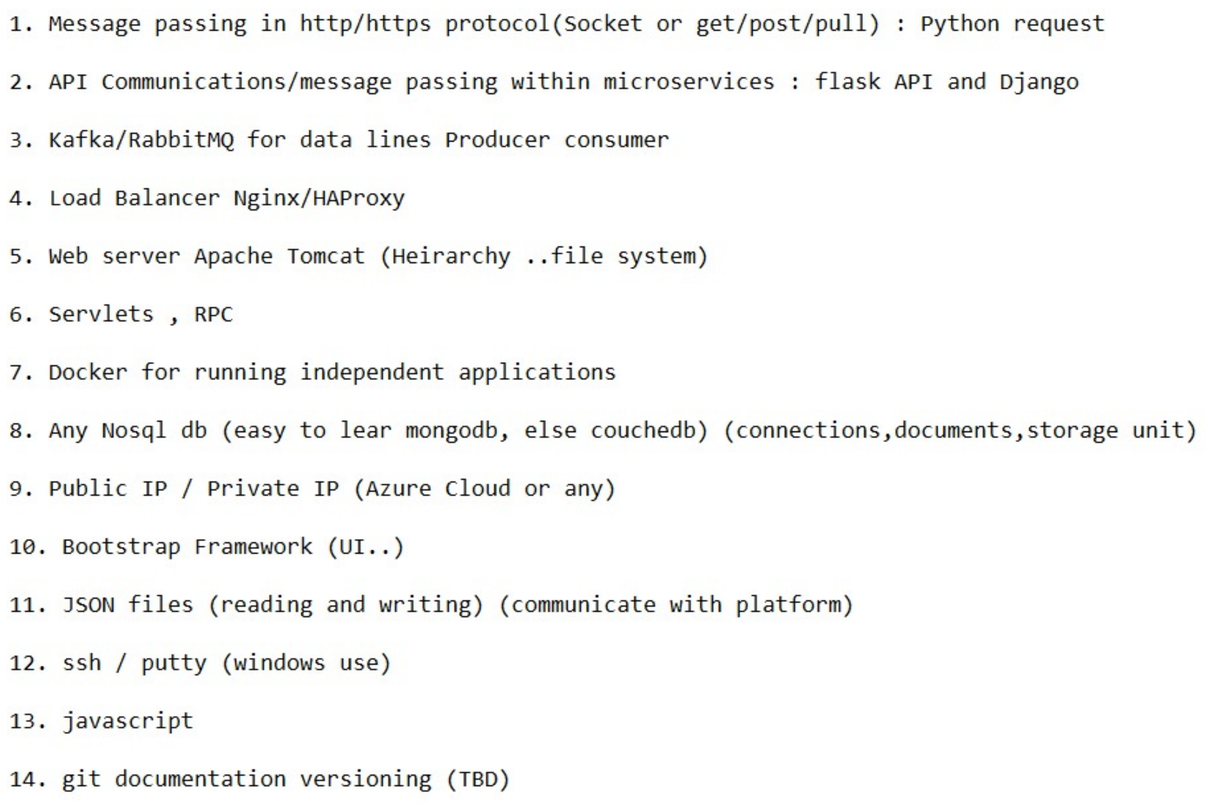


Project Talks

- Distribute work wisely among yourselves in team.
- Communicate, communicate & communicate.
- Too many cooks in Architecture design.
- Independent Work.
- Design, document & implement. Don't procrastinate.

50-60% of the project is designing and documentation. If designing is done well, implementation will be easier.

Tech Stack(credits: Pratik)



1. Message passing in http/https protocol(Socket or get/post/pull) : Python request
2. API Communications/message passing within microservices : flask API and Django
3. Kafka/RabbitMQ for data lines Producer consumer
4. Load Balancer Nginx/HAProxy
5. Web server Apache Tomcat (Heirarchy ..file system)
6. Servlets , RPC
7. Docker for running independent applications
8. Any Nosql db (easy to lear mongodb, else couchedb) (connections,documents,storage unit)
9. Public IP / Private IP (Azure Cloud or any)
10. Bootstrap Framework (UI..)
11. JSON files (reading and writing) (communicate with platform)
12. ssh / putty (windows use)
13. javascript
14. git documentation versioning (TBD)