SCP Build 1.0.0 Hour-By-Hour Plan

Justin| Lingxing | Ming Yu | Tanny | Keeve | Hong Liang

Section 5, Team 5

COO, Prof. Rafael J. Barros

Overview:

SCP build 1.0.0 has been approved by QA and our team has planned to deploy it on the 7th of February. In preparation of this deployment, our team has completed a round of pre-deployment risk analysis and configuration management. The top three risks identified are infrastructure related; of which team has devised recommendations to mitigate them. Additional risks are covered in the Annex along with the configuration management. This deployment plan was created after taking these risks into consideration.

Risk Assessment/Recommendation(TOP 3):

Priority	Description	Likelihood	Impact	Recommendation
1	Production server crashes during business hours, and totally stops working.	Medium	High	Mitigation - Run another server instance that mirrors the production server, and in the event of an unexpected crash, set AWS's Elastic IP to automatically switch over to the mirrored backup server. This process should not be noticeable for the end users (clients).
2	Peak traffic flow may exceed server load capacity, resulting in long response times/unsuccessful responses	Medium	Low	Mitigation - Set up multiple instances of the same application and use a load balancer to distribute traffic across the different instances.
3	The AMI released by the development team is incompatible with the infrastructure that the application will be deployed on	Low	Critical	Transfer - Let the developers know about the limitations and specifications of our infrastructure before they mark the development as complete so that the responsibility lies on the developers to ensure that their program can run on our infrastructure. Mitigation - Deploy on a separate testing instance to check if any problem occurs.

As most of these likely occurring/high impact risks can be mitigated by establishing cold/warm standby with load balancing, the team will work closely with COO and the Product Manager to set up a backup instance of the application. This will be planned for implementation after a successful deployment of build 1.0.0.

Hour By Hour Plan:

Date of Deployment: 7th Feb 2019
Change Number: SCP100X0219
Business Owner: Prof. Paul Griffin
Deployment Owner: Prof. Rafael J. Barros

Task ID	Date	Start Time	End Time	Task	Team	Responsible (In Charge)	Preparation	Post-Check	Dependency	Status
T1	28-Jan-19	16:00	16:30	Get AWS account registered with build product manager and cloudtopus.	Infrastructure	JC	Created AWS account	Google form submission successful, and account added successfully on cloudtopus	-	completed
T2	31-Jan-19	15:00	20:00	Complete Hour-by-Hour plan and upload on e-learn	Infrastructure	JC	Conducted team meeting and created Hour-by-Hour plan draft.	Hour-by-Hour document successfully uploaded on elearn	-	Completed
T3	1-Feb-19	18:00	21:00	Get AMI from product managers	Developers, Infrastructure	JC	1. Product manager approval on Hour-by-Hour plan 2. Build 1.0.0 completed by developers and approved by QA.	AMI shared on AWS EC2 dashboard	T1,T2	Completed
T4	7-Feb-19	16:00	16:30	Launch Deployment Environment Machine (EC2 instance) and configure security group (SSH)	Security	MY	SSH key pair generated using puttyGen	Upon Launching the instance, clicking on view Instance shows the instance running.	T3	Not started
T5	7-Feb-19	16:30	17:00	Enable connection via SSH and retrieve security credentials and public DNS of AWS EC2 instance.	Security	MY	Security groups successfully created on EC2	Security keys successfully downloaded	T4	Not started
T6	7-Feb-19	17:00	17:15	Test SSH connection via putty.	Security	MY	Security keys downloaded	Downloaded security keys able to establish connection to AWS EC2 instance successfully via putty using SSH	T5	Not started
T7	7-Feb-19	17:15	17:30	Assign Elastic IP address and test connection	Infrastructure	LX	AWS EC2 instance created	Deployment team able to connect to EC2 instance via elastic IP address	T6	Not started
T8	7-Feb-19	17:30	18:00	Test build for business functionality - smoke test - Read-only-test - Functional and full test.	Infrastructure, Quality Assurance	LX	Product Manager and COO must be available to validate deployment success via email.	1. The application functions and meets the required business c2criteria 2. All tests conducted on the application successful	T7	Not started

Note: Only the In Charge is listed in the Hour by Hour plan. The full team breakdown and contact details are as stated in the ANNEX.

ANNEX

Contacts:

Initials	Name	Phone	Email	Manager	Team
MY	Chua Ming Yu	9888 0880	mingyu.chua.2017@sis.smu.edu.sg	Justin Choy	Security
JC	Justin Choy	9321 9330	justin.choy.2017@sis.smu.edu.sg	Prof. Rafael J. Barros (COO)	Communications (IC)
TL	Tanny Lai	8533 8562	tanny.lai.2017@sis.smu.edu.sg	Justin Choy	Communications
HL	Ng Hong Liang	9854 3232	hlng.2017@sis.smu.edu.sg	Justin Choy	Quality Assurance
LX	Guo Lingxing	9271 4836	lxguo.2017@sis.smu.edu.sg	Justin Choy	Infrastructure (IC)
KQ	Keeve Quah	9658 6525	keeve.quah.2017@business.smu.edu.sg	Justin Choy	Infrastructure
WT	Wendy Tan	6828 1967	wendytanlc@smu.edu.sg	-	Developer
PR	Professor Rafael	6828 9675	rafaelbarros@smu.edu.sg	-	COO

Risk Assessment/Mitigation (FULL):

Category	Risk Description	Likelihood	Impact	Recommendation
Infrastructure	Significant portions of Amazon Web Services (AWS) experiences a downtime for an unspecified duration, resulting in unavailability of clients' access to the Smart Contracts application hosted on the platform.	Low	Critical	Mitigation -Cold/Warm Standby techniques could be used through deploying a secondary backed-up AWS EC2 instance, reconfigure domain services to point to new address.
Security	A previously unknown security vulnerability is discovered in the software underlying the web server hosting the Smart Contracts application, which may be used by bad actors to gain unauthorized access to system resources.	Low	High	Mitigation - Apply all software updates as they are made available, and remove non-essential components from the production server. Monitor security vulnerability databases such as the Common Vulnerabilities and Exposures (CVE) database [https://cve.mitre.org] to see if any of the software / systems in use are affected by any new discoveries.
Infrastructure	A software or hardware conflict/issue causes the production server to slow down or have excessive resource consumption, resulting in sluggish performance on the clients' end for the Smart Contract application.	Low	Medium	Avoidance - More effort on pre-deployment testing and quality assurance may be necessary; Research on known software / hardware combinations that may cause potential issues. Mitigation - Set up and use monitoring tools (e.g. Amazon CloudWatch, Nagios) to alert relevent personnel in the event that system resource consumption exceeds its usual boundaries. Recovery - Schedule a maintenance period during non-peak hours to restart services to see if that solves the problem. Roll-back to last working configuration if applicable.
Security	Production server's credentials and/or authentication keys are compromised.	Low	High	Mitigation - Access to production server should be granted on a need-to basis. Keep track of all user/ip logins, and resources accessed during the session. Private key to access server should only be known by the security team to prevent it from leaking. Recovery - Change passwords and update authentication keys if system has been found compromised, immediately. Further actions to be prescribed by security team.

Infrastructure	Production server crashes during business hours, and totally stops working.	Medium	High	Mitigation - Run another server instance that mirrors the production server, and in the event of an unexpected crash, set AWS's Elastic IP to automatically switch over to the mirrored backup server. This process should not be noticeable for the end users (clients).
Infrastructure	Excess charges are incurred from cloud service provider Amazon, as the Elastic IP address becomes dissociated from a running instance.	Low	Low	Mitigation - Set up and use monitoring tools (e.g. Amazon CloudWatch, Nagios) to alert relevant personnel in the event that system goes offline.
Infrastructure	Peak traffic flow to the application may exceed what the server is capable of handling, resulting in server taking longer to serve required content to customers	Medium	Low	Mitigation - Set up a multiple instance of the same application, and use a load balancer to distribute traffic across the different instances
Security	Server may come under DDOS attacks from malicious sites, resulting in compromised ability for service to actual customers	Low	High	Set up monitoring tools to monitor network traffic. Be prepared to ban IP addresses if malicious intent (eg: pinged over 50 times) is detected. Make use of captcha to detect if actual users are trying to reach the site
Infrastructure	The AMI released by the development team is incompatible with the infrastructure that we have	Low	Critical	Acceptance - We have to accept the risk. Transfer - Let the developers know about the limitations and specifications of our infrastructure before they mark the development as complete so that the responsibility lies on the developers to ensure that their program can run on our infrastructure Mitigation - Deploy on a separate testing instance to check if any problem occurs

Configuration Management:

Deployment Environment	AWS EC2
Instance type	AWS T2.micro
vCPU	1
Memory (GIB)	1
Instance Storage	EBS only
Network	vpc-9fcee6f8
Storage Size	10GB
Volumn Type	General Purpose SSD (gp2)
IOPS (Input/Output Operations Per Second)	100/3000
Encrypted	**No**

Operating System	Red/centos (Linux)
Web Server	Nginx 1:1.14.1-2.34
Web framework	Django v2.1.5
Language	Python 3.6
pip	9.0.3
Elastic IP	**<To be set during deployment>**

Montoring tools	
htop	Cloudwatch