**Foundations in Digital Development (for the Cloud)**

**Delivery Plan**

Table of Contents

[Course Information - 2 -](#_Toc394389653)

[Course Overview - 3 -](#_Toc394389654)

[Trainer Notes 4](#_Toc394389655)

[2. Course Overview 4](#_Toc394389656)

[3. Session 1 5](#_Toc394389657)

# Course Information

**Diagnosis of Need**

It has been recognized that to support the Building Performance and Systems ‘’digital/smart buildings’’ service offering that there is a need for engineers across all grades to be trained to take full advantage of new digital/cloud resources that are being made available. Further, a large proportion of research carried out last year within the BPS skills community in 2017/18 leveraged upon AWS cloud infrastructure. This appears a clear direction of travel for the discipline. AWS essentials training has been arranged for those working in the Technology Consulting UK group and this was very well attended. However, attendees at the event provided mixed feedback – citing the need for pre-requisite knowledge and a need for a more ‘’hands-on’’ experience to develop familiarity.

We believe that this experience of business needs is not constrained to these cases cited within the Building Performance and Systems skills network, but one shared to some extent with most skills networks as a means of facilitating the firms ‘’Digital Transformation’’. For example, we see an example of direction application of these skills to the ‘’Movement Insight’’ initiative to support transport infrastructure planning. Such foundations skills are internationally applicable and need only to be tailored geographically to regional language/dialect and custom.

This course is intended to provide attendees with the ‘’pre-requisite’’ experience to begin effectively engaging with digital development projects. Through this a community of engaged developers could be established within the firm to support delivery of Arup’s process automation and insight platforms.

**Learning Outcomes**

By the end of this course attendees will be able to:

* Set up a fit for purpose IDE
* Write and execute basic programs in Python
* Understand the terms “Clone”, “Push”, “Pull”, “Add”, “Commit”, “Merge”, “Checkout” and “Revert” in reference to the use of Version Control Systems and gain experience in making a selection of these requests.
* Clone and make changes to a pre-prepared HTML website
* Log in to AWS
* Set up security groups and access policies.
* Upload content to S3
* Launch an EC2 instance, work with AMI’s and establish secure connections.
* Make requests to a REST API
* Read JSON and work with the Response Payload.
* Create and deploy a simple web-application.

**Audience**

Global audience from any discipline and grade

**Class length**

xx

**Number of attendees**

12 – 14 participants

**Materials & Room Setup**

* Participant laptops
* Flipchart
* PowerPoint
* AWS account for course with all participants registered as users
* AWS workspace AMI (for those participants who have trouble completing pre-work activities)
* Elsys sensor and gateway (optional)
* Access to Nyquist-dev network
* Git, AWS CLI and Bash cheat sheets.

**Pre-course work**

* install git
* install python3
* install Atom text editor
* install putty
* install Docker Desktop
* install AWS CLI
* create an account on Github
* create an account with Docker Hub

# 

# Course Overview

|  |  |
| --- | --- |
| Timings | Session |
| **09:00** | **Course Overview and Introduction** |
| **09:30** | **Development EcoSystem** |
| 12:15 | *Lunch break* |
| **12:50** | **Cloud Services** |
| 14:15 | *Welfare break* |
| **14:30** | **Interfaces** |
| **17:00** | **Recap** |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

# Trine Notes

|  |  |
| --- | --- |
| **Session Name** | Course Overview and Introduction |
| **Time** | 09:00 |
| **Trainer** | Internal Facilitator |
| **Learning Outcomes** | * View the structure of the day * View the main course topics and how they relate to one another |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time** | **Duration** | **Topic** | **Delivery Method** | **Materials** |
| 09:00 | 10 mins | Course Overview | TRAINERS to welcome group and introduce themselves  Ensure each attendee has created an account on Bitbucket.  **Laptops**  Ensure each attendee has a laptop and has downloaded the following software as part of the course pre-work:   * git * python3 * Atom * Putty * Docker Desktop * AWS CLI   If participants have trouble completing this pre-work we can provide them with AWS Workspace. However, this workspace will be hosted online and won’t be available to them after the course finishes, therefore this is just a fallback option to enable the course to continue in a timely manner. | * Git downloads: <https://git-scm.com/downloads> * Python downloads: <https://www.python.org/downloads/> * Atom Text Editor: <https://atom.io> * Putty downloads: <https://www.putty.org/> * Github: <https://github.com/> * Docker Desktop: [https://docs.docker.com/v17.09/engine/installation/#supported-platforms](https://docs.docker.com/v17.09/engine/installation/" \l "supported-platforms) * Install AWS CLI: <https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html> |
| 09:10 | 10 mins | Icebreaker | ASK Participants to think about inventing one thing that would have the biggest impact at your work at Arup.  TRAINERS to share what they would want to invent and ASK participants to take two minutes to explain their invention and the impact it could have.  ASK for volunteers to introduce themselves (background/office) and then share their idea. Go around the class until all participants have shared.  Trainers to record any interesting ideas on a flipchart to potentially cover later. |  |
| 09:20 | 10 mins | Course Overview and Learning Outcomes | EXPLAIN:   * Course learning outcomes * Agenda blocks & flow * Confirm tea breaks, lunch and finish time.   INTRODUCE Process Map and how this relates to the course – this is how the course fits together  ASK if anyone has any previous experience of any of the elements on the Process Map. This course will tend to assume that no one has previous experience. |  |

**Design Notes**

|  |  |
| --- | --- |
| **Session Name** | Development EcoSystem |
| **Time** | 09:30 |
| **Trainer** | Internal Facilitator |
| **Learning Outcomes** | * Install and configure dependent tools. * Explain what code is and the names of some popular programming languages * First use of an IDE (Integrated Development Environment) is and familiarity with Atom user interface * Write a simple python program * Install and use code libraries/packages/modules are and how they provide pre-developed functionality * Explain VCS (Version Control Systems) and how developers use them to collaborate on and share code * Explain the difference between a local and a remote repository and the basics of branching * Use basic git commands to interact with the Github VCS to clone a repository, make changes in a local branch and push changes back to a new branch on the remote repository |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time** | **Duration** | **Topic** | **Delivery Method** | **Materials** |
| 09:30 | 10 mins | Overview of code and language support | EXPLAIN that the purpose of this session is to:   * Introduce “The Development Ecosystem” topic on the Process Map   + Code and language support   + Intelligent development environments   + Syntax highlighting and debugging   + First-steps in programming   + Libraries, packages and refactoring   + Version Control Systems   STATE that there are different types of programming language and each is used for different purposes.  INTRODUCE HTML, Javascript, Python as currently popular languages – for awareness. Point out their different purposes and that depending on the project you may need to use multiple languages. |  |
| 09:40 | 40 mins | Syntax highlighting and debugging | TRAINER to demonstrate example of Python code written incorrectly in Notepad with no error checking, and then same code written incorrectly in Atom where errors are highlighted.  INTRODUCE what an IDE is and how IDEs help developers write code. Point out that there are different IDEs for different programming languages.  EXPLAIN that an IDE will understand what code is being written and is able to pick up mistakes and suggest how to complete code segments.  STATE that today we will be using Atom as our IDE and give brief overview.  ACTIVITY – everyone to install Git, Python, Atom and Putty if they haven’t already. Set-up Python language support for Atom-IDE, followed by HTML language support and Jupyter Notebook file support. – 15 minutes  ACTIVITY – everyone to open Atom and follow trainer instructions to explore the main parts of the UI together as a class. (Live Demo) 5 mins. Trainer to introduce project folder, file management, file and terminal views. | * Git downloads: <https://git-scm.com/downloads> * Python downloads: <https://www.python.org/downloads/> * Atom Text Editor: <https://atom.io> * Putty downloads: <https://www.putty.org/> * Instructions to set-up Python language support for Atom IDE: <https://atom.io/packages/ide-python> * Instructions to set-up HTML language support for Atom IDE <https://atom.io/packages/ide-html> * Instructions to set-up Jupyter Notebook file support for Atom IDE <https://atom.io/packages/jupyter-notebook> |
| 10:20 | 30 mins | Programming | STATE that we will be using Python code and will write our first Python program in Atom.  EXPLAIN the basics of Python:   * 2 versions - Python 2 and Python 3 * Python uses indents to format code instead of brackets   ACTIVITY - Building on the installation of the IDE, participants will write code that prints ‘Hello World’ using Python and learn how to run code using command line. (Step by Step) 15 mins | Code Academy Python courses provides more details on Python development: <https://www.codecademy.com/catalog/language/python> |
| 10:50 | 30 mins | Libraries, Packages and Refactoring | INTRODUCE the concept of code libraries/packages/modules and how they provide pre-developed functionality, so you don’t have to write everything from scratch every time.  ACTIVITY – Building on previous activity, change ‘Hello world’ program to also print the current date. Show how code fails initially due to missing datetime module. Change code to import module and then see how code runs correctly. (Step by Step) 15 mins | Link to list of Python modules: <https://docs.python.org/3/py-modindex.html> |
| 11:20 | 55 mins | Version Control System | EXPLAIN that:   * very often multiple people will collaborate on the same piece of code. * people want to share code and collaborate on code without overwriting each other’s changes. Give example of all the developers at Google working together to write code for web search.   INTRODUCE Version Control Systems (VCS) as a means of sharing and collaborating on code with thousands of other people without overwriting others’ changes.  STATE that there are many different VCS – Github, Gitlab, Bitbucket – all hosted on the internet. Explain that Arup has its own version of Gitlab which all Arup employees can log into with their Arup credentials that can only be accessed within the Arup VPN (those using AWS Workspace will not be able to access this).  ACTIVITY – everyone to navigate to github.com and log in using their credentials. Attendees to share Github username with Trainers so that they can be added to the course repository. Attendees to look at shared repository of code and view course material. (Live demo) 5 mins  INTRODUCE Git – tool to communicate between the VCS repo and the local laptop. Most VCS use Git including all those mentioned above. Git commands let you get code from the repo onto your laptop and push changes you have made on your laptop back up to the online repo, plus lots of other things.  PRESENT diagram of a generic VCS including *remote repositories* on Gitlab, Github, Bitbucket cloud, *local repositories* on local laptops/machines, and how Git sits in the middle. We will explore the most used commands – git clone, git add, git commit, git pull, git push.  INTRODUCE branching and how to use branches to make changes to repository code. Show overview diagram of simple branching pattern and explain that everyone currently has a copy of the ‘master branch’ of the course repo.  ACTIVITY – Git clone: everyone to clone the course repository onto their local laptop with the ‘git clone’ command. Everyone should now have a local copy of all the course code on their laptops. View in file explorer. Then open new Atom project from local copy of cloned course repo. (Step by Step) 15 mins  ACTIVITY – Participants to create a new local branch on their laptops and add their ‘hello world’ python script into an existing ‘helloworld’ directory in their local repo. Participants to push their local repo branch to the remote repo using the git commands add, commit, pull and push. Participants can then view new branches appearing on the remote Github repo (Step by Step) 10 mins and make a git “fetch” request to make the latest changes available to “checkout” or “pull’ onto their local machine.  EXPLAIN how participants can make changes to someone else’s code and everyone can edit the same file.  ACTIVITY – Building on previous activity, participants to make changes to “my-first-page.html” and push to their branch in the remote repo. View pre-prepared HTML code in browser on laptop at <http://localhost>, make changes to the code using Atom, visualize changes in browser on laptop at <http://localhost>. Add, commit and push changes from local branch on laptop to remote branch on Bitbucket repo. At the end of the session, those who have completed early may “fetch” the latest updates from the remote and “checkout” a remote branch created by someone else on their local to view at <http://localhost> (Step by Step Guide) (30 mins)  Provide additional details on how to use git revert in case of mistakes – only for those who completed all other activities early.  ASK – Do any of you (class members) think that a Git Version Control System could be employed on any of your current projects to keep track of a shared set of files? | * Git commands cheat sheet: <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf> * Arup Gitlab: <https://gitlab.arup.com/> * Github: <https://github.com/> * Course repository: <https://github.com/ArupAus/digital-foundations-cloud.git> |

|  |  |
| --- | --- |
| **Session Name** | Cloud Services |
| **Time** | 12:50 |
| **Trainer** | Internal Facilitator |
| **Learning Outcomes** | * Explain what the ‘cloud’ is and that there are various cloud providers * Explain what the basic elements of cloud service are including storage, compute, database, security, distribution… * Identify AWS basic services and use the AWS service console * Explain what an S3 bucket is and use to serve websites * Explain what an EC2 instance is and use to serve websites |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time** | **Duration** | **Topic** | **Delivery Method** | **Materials** |
| 12:50 | 30 mins | Introduction | EXPLAIN that the purpose of this session is to:   * Introduce “Cloud Services” on the Process Map   + Overview of providers and typical services available   + Compute and storage   + Security and network protocols   INTRODUCE the cloud and what it means. Give overview of various cloud providers and how they generally all provide some typical components: storage, compute, …  STATE that we will be focusing on AWS for the rest of the course and describe the following AWS in more detail:   * Compute * Storage * Database * Access Management * Security * Distribution * Management and Governance   ACTIVITY - Log into AWS console and configure AWS CLI. Explore the interface focusing on the AWS services introduced above (Live demo) 5 mins | * Arup AWS federated account login: [aws.arup.com](file:///Volumes/JOBS/260000/263642-50 Digital Foundations/3 Internal/3.1 Course content/aws.arup.com) |
| 13:20 | 65 mins | Compute/Storage  Security/Networking Protocols | ACTIVITY – Create an S3 bucket and upload (drag and drop) pre-prepared website files from local version of course repo using S3 user interface. Configure bucket policy to allow public access and view via URL. (Step by Step) 20 mins  ACTIVITY – Create Linux EC2 instance (free tier) and log in using credentials and putty. Explore directory structure (same as any Linux machine) using command line. Install python and install git using Linux yum commands. Clone course repository onto EC2 instance and checkout participant’s own branch. Run python code on EC2, just the same as how it is run on local laptop. Install and start the LAMP Web Server on the EC2 instance. Copy “my-first-page.html” from EC2 git project directory onto EC2 instance directory “/var/www/html”. View on the machines public IP address. (Step by Step) 45 mins.  Trainers will demonstrate this process step-by-step at the same time as each member of the class follows themselves. | Setting AWS S3 Bucket Policy: <https://docs.aws.amazon.com/AmazonS3/latest/user-guide/add-bucket-policy.html>  Linux command line cheat sheet: <https://files.fosswire.com/2007/08/fwunixref.pdf>  Install a LAMP Web Server with the Amazon Linux AMI: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html> |
| 14:25 | 5 mins | Network security question | EXPLAIN - that content they have created is being distributed over an unsecured protocol (HTTP) and hence traffic is unencrypted and may be intercepted. This means the page may not be viewed from some networks or be retrieved from popular search engines.  ASK - The web content you have made available is accessed using `HTTP` protocol. Why shouldn’t we trust this web content? How could we ensure that such web content can be trusted?  Answer = apply an SSL certificate and view content on port 443 (HTTPS). |  |

**Design Notes**

XXX

|  |  |
| --- | --- |
| **Session Name** | **Interfaces** |
| **Time** | 14:30 |
| **Trainer** | Internal Facilitator |
| **Learning Outcomes** | * Explain common types of API’s and how they could be used. * Make requests to a REST API. * Read JSON and work with the Response Payload. * Create and deploy a simple web-application. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time** | **Duration** | **Topic** | **Delivery Method** | **Materials** |
| 14:30 | 20mins | An introduction to Application Programmatic Interfaces (API’s) | EXPLAIN that the purpose of this session is to:   * Introduce “Interfaces” on the Process Map   + Application programmatic interfaces   + Making simple API requests   + Working with response payloads and understanding JSON   + Create and deploy a simple web application   QUESTION – Ask the class: do applications need store and manage all dependent data themselves? Answer = no, often we make requests to other party’s resources.  INTRODUCE – Explain how there is often no need to develop and manage all the dependent data sources of an application. It is common to make requests to other parties to retrieve or send information to enable applications by working with Application Programming Interfaces. Sometimes such requests require authentication.  EXPLAIN – Three types of Application Programming Interfaces using example web services and key features with diagrams:   * BACnet Protocol (HVAC Automation): SOAP API * Google Maps Geocoding Service: REST API * WhatsApp: Websocket API |  |
| 14:50 | 20 mins | Making simple API requests | ACTIVITY – Each member of the class is to make a RESTful GET request to an Arup Elasticsearch domain storing data collected by a wireless environmental sensor located in the classroom using the “my-first-request.ipnyb” Jupyter Notebook. The intention is for them to retrieve the most recent record and print the result.  ASK – members of the class to volunteer an explanation of each variable within the JSON response payload and how to query parts of it. | Gateway/  Sensor  <https://search-lorawan-test-elasticsearch-pjnzn3uxajzeecqr2fberl2pa4.eu-west-1.es.amazonaws.com/> |
| 15:10 | 45 mins | Working with the response payload and understanding JSON | ACTIVITY – Each member of the class will now make further RESTful GET requests by editing the Jupyter Notebook. Firstly, to retrieve just the `temperature` result of the most recent record and print the result. Secondly, to retrieve a 10-minute period of time-stamped `temperature` results and print as a pandas dataframe. |  |
| 15:55 | 65 mins | Create and deploy a simple web application | ACTIVITY – Each class member is to create a Docker image of the pre-prepared Flask web application in the course repository by following the instructions in the README.md file. They will then tag the Docker image and push to AWS ECR using AWS CLI. Following this, each class member will start a Fargate Cluster using AWS ECS, launch an Application Load Balancer and create a Target Group. They will then launch and AWS ECS Service and distribute over Cloudfront. Those who complete the task may interact with the service at [resource].cloudfront.net.  Trainers will demonstrate this process step-by-step at the same time as each member of the class follows themselves. | ECR Docker basics:  <https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-basics.html#use-ecr>  Create Application Load Balancer:  <https://docs.aws.amazon.com/AmazonECS/latest/userguide/create-application-load-balancer.html>  Launch a service on ECS:  <https://docs.aws.amazon.com/AmazonECS/latest/userguide/create-service.html>  Distribute service over Cloudfront:  <https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-working-with.html> |

**Design Notes**

XXX

|  |  |
| --- | --- |
| **Session Name** | **Recap** |
| **Time** | 17:00 |
| **Trainer** | Internal facilitator |
| **Learning Outcomes** | * Retrieve guidance and resources post-course to continue engagement with topic. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time** | **Duration** | **Topic** | **Delivery Method** | **Materials** |
| 17:00 | 15 mins | Takeaway’s | ASK each individual to spend a few minutes to think of 2 key things they take away from the course.  ASK each individual to share within pairs |  |
| 17:15 | 5 mins | Summary | REVIEW the days agenda and RECAP the learning outcomes |  |
| 17:20 | 10 mins | Next Steps | Discuss any post-work, any handouts/reference guides that can be referred to after the course. | <https://aws.amazon.com/training/learning-paths/>  <https://www.codecademy.com/catalog/language/python>  <https://www.codecademy.com/catalog/language/javascript> |
| 17:30 |  | Close | TRAINERS to thank participants and close the day |  |
| Next day | 10 mins | Evaluate | Send participants an online questionnaire to evaluate course successes and areas for improvement | Online questionnaire |