So you want to be a successful engineer/developer

Tags: ACG

At A Cloud Guru, we have students coming from a range of backgrounds. Many come to cloud computing knowing only that they want to have a profitable career but not understanding what is needed to achieve that success. The information below has been collated to help those students who desire a successful career and want to develop some of the skills to convert knowledge & a job to a respected career.

So you want to be an Engineer or Developer

You may not have thought of it in those words. You may have thought I want a well paid job I can get into quickly, or I want a change to job where I can solve problems and have people look up to me or even I don't know what to do with my life but someone said that I could be good at software development or devops.

How you got here is less important than understanding what this is and what you need to do to achieve your goals. If you want good money, promotion and recognition you need to prepare yourself to be a good engineer. These are learned skills that take time and practice to learn, so you need to start now.

- With these skills you will quickly be seen as an asset and offered more interesting assignments and career progression.
- Without these skills the risk is that you will never be able to do more than the simplest task. Never contribute insight and innovation, and will be nuisance to your peers when you constantly ask then to do your thinking for you.
- Without these skills you might also find that your job will be replaced by some automation that does the simple tasks faster, more reliably, and for no real cost. Wouldn't it be sad if a piece of machine learning software could do your job better, simply because you had not learned higher skills.

So what makes a good engineer or developer?

It does not matter which discipline you choose, the skills are the same.

Preempt problems

Work with your eyes open and always questioning what are the possibilities, and the limits, and risks. Design to avoid problems.

Seek truth & answers voluntarily

This means constantly learning. Seek out authoritative information and the reasons beneath the rules. Verify what you think you know, and validate what others tell you. (Be careful as there is more misinformation on the internet than correct knowledge).

Communicate clearly

Effective communication is where you provide the other person with sufficient information for them to understand the situation. Communicate the issue, the context (exactly where you are experiencing it), your perspective, and the feedback you are seeking. Use open language that invites discussion. Write full sentences, not just words. Write paragraphs, not isolated sentences. Paint a picture for people that shows how you have examined a subject from multiple angles (This is basically the opposite of Txt speak).

Support others

Encourage the input of others by polite questioning and recognition & confirmation when their logic is good. Even when we disagree with others, we can learn by trying to understand their perspective and why they reached their point of view. You may have missed something, or they may have missed something, but by being supporting and encouraging, we will achieve a better combined result.

Decide to become a good engineer.

Being a good engineer is a learned behavior. It takes discipline and practice, and starts now.

It is worth noting that even though we teach on-line and in an exam focused manner, we do not reject or devalue the skill that traditional academic classroom training offers. In a traditional engineering school, much of the 1st year of classes are about teaching foundation questioning and research skills. Do not think of these skills as irrelevant or a waste of time. Once you have technical skills and a job, the key to progress and promotion is these higher skills.

So what does all this mean in practical terms ?

Positive skills and behaviors that you need to be developed:

1. Ask yourself questions constantly.
2. Phrase the questions you ask yourself for clarity. Break the question into small pieces and steps until you get the the core of the issue.
3. Ask yourself where authoritative information can be found. Not opinion or vague sloppy description, but solid fact based on knowledge and research.
4. Seek initial understanding yourself. Seek support information from the environment (logs, errors, the results produced), check information that you have been provided with, do research, do some experimentation.
5. Validate your answers (or questions) in discussions with others. Communicate clearly. Often you will find the answer you need simply by explaining the problem to others (and yourself).
6. Listen to others. Seek to understand their different approach and point of view. Often we become stuck because our understanding is wrong. So be open to the views of others to see if they show you a perspective you had missed.
7. Help others. Use open and clear communication to help others solve problems. There is much to be learned from each others' problems. Maybe by helping them you will avoid a problem yourself, or have an insight that lifts you to a new level of understanding, or makes a friend and contact that will help you build a career.

You will note that the 1st four are to yourself. You must start with an internal dialogue. This is where you grow your engineering skills. Technical knowledge is just the raw material that you use to convert insights to results.

These are learned skills. So start now and remind yourself that taking the easy route may not help you be a better engineer, developer or eventually engineering manager.

Example 1:

Problem- You want to ensure your data on a Webstore is safe but do not understand how the backup works?

The questions an engineer asks themselves:

- What am I concerned about; backups or data protection?
- Protection from what? Loss, Corruption, Theft, Misuse, ...
- How much protection do I want ? What is the true value to the company? What will the company lose if one of these things happen?
- How many options and angles do I have to debug this?
- What does the vendor offer as standard or as an add-on? How does it address each of the concerns you have identified as important?
- Which of these can be implement within the bounds of the value to the company?
- With what I now know, is a Webstore the right storage solution?

The questions an engineer asks others:

- Seek opinion of others about gaps between what you need and what you have available to you.
- Seek opinion of others about experiences or insights in using the solutions you have tentatively selected.
- Are there other approaches that might be considered?

Example 2:

Problem - My lab does not work!

The questions an engineers asks themselves:

- What information do I have? How do I know it failed?
- What was the last step that it was working?
- What error messages do I have, or can I find?
- What vendor, or credible source, information is there about the error message(s)?
- Have I checked the syntax, and characters used?
- Have I checked the way I did the lab, and does it match the instructions?
- How can I adapt what I have to allow the lab to be successful?

The questions an engineer asks of others:

- How could this option be used in production?
- How do I interpret this code in the context of this documentation?
- Is there a more elegant/efficient way of doing this?
- What am I missing, this does not seem logical?
