You don’t have to code/configure/maintain it yourself

(an opinion from a tired do-it-yourself programmer)

Use Free/Open Source Software

- Choose high quality, proven, and supported free/open source software (when applicable)
  - It’s not all about the licensing cost
  - It’s not always the best solution, too
  - However, many of them are very good, proven, and used by millions
- You can freely use, modify, redistribute, and sell it (please, read and understand the license)
  - However, you *may* need to understand the software first

Use hosted services (1)

What about Uptime & Monitoring?

- Email:
  - It’s not about installation and configuration
  - When a lot of users are using your email servers, in mixed and uncontrolled environments: you have to do it right
  - Sometimes, it is easier to use a hosted and proven email services (at your own domain name; however, your emails are not in your complete control)
- Website:
  - You may code and host it yourself, or you can use a proven software and/or hosted service
  - Think about web standards, mobile devices, and many more

Use hosted services (2)

- Customer service and/or support ticket
- Cloud:
  - IaaS
  - PaaS
  - SaaS
  - DBaaS
  - (and more)

Monitoring

- More uptime is good
- Downtime is not good
- You may run a monitoring service yourself, but who monitors this service?
- Consider outsource monitoring service to some dedicated companies
  - They may also monitor for performance

Version control (1)

- So, you decide to code it yourself
- Now, you *have to* manage your source code
- And, it’s not all about the backup (so you don’t lose the codes)
  - How about several programmers are working on the same function in the same file, in your current source code tree?
  - How about untested branch of your codes?
  - How about rollback to old version?

Version control (2)

- If you do the source code management yourself, you surely will miss features offered by modern source code management or version control system
  - Some of them are available as hosted solution (so you don’t have to run a dedicated server yourself, and rely on them for the backup)

Don’t Repeat Yourself (1)

- DRY vs WET
- Standards available
- File and data-interchange format
  - Are you really sure that you have to invent a new file or data-interchange format? What about common used/standard file format and/or data-interchange? What about lightweight embedded database system?
- Modeling
  - Some modeling languages are not easy to learn
  - However, unless you are expert in this field, it is better to learn and use the standard ones
  - Think about new team member and external parties

Don’t Repeat Yourself (2)

- Architecture
  - You can design whatever you want
  - Until you have to manage it for years, or you have a new team member, or you see the limitation of current design
  - Monolithic vs microservices
- Libraries

Do your homework

- If you understand the underpinning theory, you may not spend too much time on one feature/function
  - You don’t have to research it yourself

Testing and Continuous Integration

- Testing – even before writing the codes – is very important
  - You don’t have to make your own testing method
  - You don’t have to write your own tool/library
- Continuous integration is also very important if your are not working alone or your software is relatively complex
  - Hosted services are available

Thank you