## Edge Platform & Ingress Coding Challenge

Members of the Edge Platform & Ingress (EPI) team are responsible for development, operations and configuration management of EPI's systems. Core competencies include:

1. Programming (Go, Python, Bash)
2. Configuration management (We use Ansible & Salt, Puppet & Chef are also applicable)
3. Networking and REST fundamentals
4. Automation - scripting, *NIX power tools, etc.
5. Documentation/Excellence
6. Having fun!

The following problem is an opportunity to demonstrate your knowledge in these areas and perhaps learn something along the way.

Your Challenge:

1. **Programming:** Write a program in Go/Python called *replayd*; this program is meant to be used as a configurable HTTP test back-end. The *replayd* application accepts user data over HTTP and stores it to an in-memory buffer. It also accepts requests for that data over HTTP and responds with what has been buffered.
    1. Use TOML, JSON, INI, etc for the configuration file format
    2. Should expose a single HTTP endpoint: /
        1. PUT or POST should store the response payload in the buffer
        2. GET should return the buffered data
    3. Your program should have unit tests
    4. Your program should be free of race conditions; one user may be setting a new payload while others are requesting it. (For example, you should be able to build your program with a race flag and run a benchmark tool against it, while manually setting new payloads)
2. **Configuration management:** Install your program
    1. Targeting the latest Ubuntu LTS release, write the necessary configuration management (salt/ansible) so that when your machine is spun up, replayd will be installed as a daemon (starts at startup, runs in background)
    2. You should create a properly locked down service user called replayd to run the deamon under.
    3. You should follow proper Linux file-system hierarchy: For example, configuration files should live in /etc/replayd, the binaries in /usr/local/bin etc.
3. **Bash scripting:** Tools to run your program
    1. Write two quality bash scripts to run your program
        1. ./replay.bash <hostname> should use curl to GET /
        2. ./set-replay.bash <hostname> < "payload" > should PUT the payload to /
4. **Automation:** Automate your deployment
    1. Putting your *replayd* project on GitHub (or equivalent)
    2. Provide 1-3 simple commands that will download, prep, compile & execute your code on at least a Mac OS or Linux equivalent. Feel free to make use of tools like vagrant, docker, etc.
5. **Documentation/Excellence**:
    1. Write a README that contains basic information to the challenge reviewer about your program/tools and instructs challenge reviewer how to, in automated fashion, use your program, e.g., curl <yourgithubrepo> | bash …. – which should handle anything challenge reviewer needs to be ready to evaluate your program.
6. **Have fun!** Feel free to add your own features and ideas that complement and extend the required features. This is your chance to show off. The more automation you put into your challenge, the better. Emphasis placed on best practices, code cleanliness, standards, etc.

**Note:** Regarding external dependencies, feel free to use them so long as they are "get-able". For example, we expect you not to write your own JSON, TOML or INI parser, but instead to use one of the popular open-source or built-in libraries.

*NOTE: While Go is the preferred programming language, you are free to write your main program in another language like Python, etc.*