Advanced topics

- Custom domain:
  - Contact the IT pro
  - Ask us (or your supporting unit reseller) to park the domain

- Working with design/development vendors (SS1, Pixo)

- Scheduling Tasks
  - guided interface to set up your cron jobs

- Error Pages
  - provides an interface for customizing the pages users see for 400, 401, 403, 404, and 500 errors

- Metrics
  - awstats, analog, and log analyzer are available for web stats, but they are disabled by default
  - Use metrics editor to enable one of the available log analyzers
  - This section also has tools to access your logs

- SSL/TLS
  - for installing certificates and generating certificate signing requests
  - SSL/TLS status
    - reports status of your certificate but also will allow you to force run auto-ssl if you need to

- Redirects

- Select PHP Version

- MultiPHP INI Editor

Managing dev, test, and prod accounts with Ansible

One way to have different cPanel development environments (dev, test, and prod) is to create a separate account for each. This isolates the environments from each other and makes restoring only one of the environments from a backup easier. There are many settings you can change in cPanel, so how do you make sure your accounts are synchronized so that your dev is an accurate environment for your prod? The cPanel team has been working on a possible way to use Ansible to accomplish this.

Ansible cpanel_api module

Everything you can customize in the cPanel Web interface is also available as a web API and the system provides a convenient command line utilities to interact with it. They are named uapi, cpapi1, cpapi2, and cpapi3 and can output results in JSON format. The cPanel team has developed a cpanel_api module to simplify interaction with the command line utilities.

https://github.com/sbutler/itpf-sp2018-cpanel-ansible

You can use the cpanel_api module to call API functions and register the results. It takes this form:

```
- name: a task that uses cpanel_api
  cpanel_api:
    version: cpapi2
    module: APIModuleName
    function: APIFunctionName
    args:
      key1: value1
      key2: value2
    register: api_result
```

The output from the cpanel_api module contains these keys:

- `result` - the value contained in the `result` or `cpanelresult` key of the output.
- `data`: the value contained in the `data` key of the result, if present.
- `stdout`: the raw value from standard output.
- `rc`: return code of the API command. This value does not indicate a success or failure of the function called.

**Example: Creating Database**

Each application in an account probably needs its own database. You can use a combination of `cpapi2` and `uapi` calls to create the database automatically.

**Wrong Way to Create Database**

```
- name: wrong way to create a cpanel database
  mysql_db:
    name: myaccount_myapp
    state: present
```

**Create Database**

```
- name: list databases
  cpanel_api:
    version: cpapi2
    module: MysqlFE
    function: listdbs
    register: db_list
- name: create webapp database
  cpanel_api:
    version: uapi
    module: Mysql
    function: create_database
    args:
      name: myaccount_myapp
    register: api_result
    when: "'myaccount_myapp' not in db_list.data|json_query('[].db')"
```

Explanation of the tasks:

- `list databases`: generate a list of all the databases for the account. This list is required so that you can tell Ansible to skip the creation if the database already exists.
- `create webapp database`: create the database if it doesn't already exist. When you name a database we require that the cPanel account name is used as a prefix. The "when" condition is what checks if the database exists, using the result from the "list the databases" task.

**Example: Creating Database User**

It's a good practice for each application to have its own database user with limited permissions. You can use more `cpapi` and `uapi` calls to create the user and assign it privileges.
Create Database User

- name: list database users
cpanel_api:
  version: cpapi2
  module: MysqlIFE
  function: listusers
  register: dbusers_list

- name: create database user
cpanel_api:
  version: uapi
  module: Mysql
  function: create_user
  args:
    - name: myaccount_myapp
    - password: changeme
  when: "'myaccount_myapp' not in dbusers_list.data|json_query('[]].user')"

- name: list database user privileges
cpanel_api:
  version: uapi
  module: Mysql
  function: get_privileges_on_database
  args:
    - user: myaccount_myapp
    - database: myaccount_myapp
  register: dbprivs_list

- name: grant database permissions to user
cpanel_api:
  version: uapi
  module: Mysql
  function: set_privileges_on_database
  args:
    - user: myaccount_myapp
    - database: myaccount_myapp
    - privileges: '{{ db_privileges|join("","\") }}'
  when: db_privileges|sort != dbprivs_list.data|sort

Explanation of the tasks:

- **list database users**: generate a list of all the database users in the account. This is required so we can skip creating the user if they already exist.
- **create database user**: create the database user if they don't already exist. When you name a database user we require that the cPanel account name is used as a prefix. The "when" condition is what checks if the database user exists, using the result from "list database users".
- **list database user privileges**: generate a list of all the database user privileges in the account. We must run this after the user is created or the user will not be present in the list.
- **grant database permissions to user**: take a list of database permissions stored in "db_privileges" and set them on the database, if the privileges don't already match what the database user has. In the "when" condition the privileges are sorted so that the order doesn't cause us to set them unnecessarily.

Example: Changing PHP Version

One thing you might want to do is run different versions of PHP for different accounts. This would let you test newer PHP versions on dev and test and then set it later on prod.
Set PHP Version

- name: list vhost php versions
  cpanel_api:
    version: uapi
    module: LangPHP
    function: php_get_vhost_versions
  register: phpvhost_list

- name: set vhost php version
  cpanel_api:
    version: uapi
    module: LangPHP
    function: php_set_vhost_versions
  args:
    vhost: myaccount.web.illinois.edu
    version: ea-php56
  when: "'ea-php56' != (phpvhost_list.data|json_query('?[vhost==`myaccount.web.illinois.edu`]')[0]).version"

Explanation of the tasks:
- **list vhost php versions**: generate a list of all the vhosts in the account and their PHP versions. PHP version settings are done by vhost name.
- **set vhost php version**: set the vhost PHP version if it isn't already set to that value. The "when" condition finds the vhost we care about in the vhost list, gets its version of PHP, and compares that to the desired version.

**cPanel API Links**

Not all things listed in the API documentation will be available for your account because not all features are enabled.

- **UAPI** - the preferred API version to use. If a function is available in this API then use it before using cPanel API 2.
- **cPanel API 2** - the older version of the API. It is more complete than UAPI.

**Troubleshooting**

- Troubleshooting customer issues
  - Regular web troubleshooting stuff
  - 503 errors