nodef / heroku-addonpool Public

Code   Pull requests   Actions   Projects   Wiki   Security   Insights

master

heroku-addonpool / index.js / <> Jump to

wolfram77 fix app name requirement ✗   History

1 contributor

127 lines (114 sloc) | 3.46 KB

```
'use strict';
const cp = require('child_process');
const _camel         = require('lodash.camelcase');
const herokuCliSetup = require('heroku-clisetup');

const RAPP    = /^[\w-]+$/;
const OPTIONS = {
  config: /\S*/g,
  log:    false
};



function HerokuAddonPool(id, app, o) {
  const unused  = [];
  const supply  = new Map();
  const removed = new Map();
  const pending = new Map();
  var o = Object.assign({}, OPTIONS, o);
  if(!RAPP.test(app)) throw new Error('Bad app name');
  herokuCliSetup();

  function log(msg) {
    if(o.log) console.log(`${id}.${msg}`);
  }

  function supplySetOne(cfg) {
    return new Promise((fres, frej) => {
      const w = cfg.split('=');
      if(w[0].search(o.config)<0) return;
      const key = w[0].substring(0, w[0].length-4);
      const val = {'value': w[1].substring(1, w[1].length-1)};
      cp.exec(`~/heroku addons:info ${key} --app ${app}`, (err, stdout) => {
        if(err) return frej(err);
        for(var r of stdout.toString().match(/[^\r\n]+/g)) {
          var k = _camel(r.startsWith('=')? 'name' : r.split(':')[0]);
          val[k] = r.substring(r.match(/[\S\s]+(=|:)\s+/g)[0].length);
        }
        supply.set(key, val);
        fres(key);
      });
    });
  }

  function supplySet() {
    return new Promise((fres, frej) => {
      cp.exec(`~/heroku config -s --app ${app}`, (err, stdout) => {
        if(err) return frej(err);
        var pro = Promise.resolve();
        for(var cfg of stdout.toString().match(/[^\r\n]+/g)||[])
          ((val) => pro = pro.then(() => supplySetOne(val)))(cfg);
        pro.then(() => fres(supply));
      });
    });
  }

  function setup() {
    log(`setup()`);
    return supplySet().then((ans) => {
      for(var key of supply.keys()) {
        log(`setup:addUnused(${key})`);
        unused.push(key);
      }
      return ans;
    });
  }

  function remove(ref) {
    return new Promise((fres) => {
      if(unused.length===0) {
        log(`remove:addPending(${ref})`);
        return pending.set(ref, fres);
      }
      const key = unused.shift();
      removed.set(ref, key);
      log(`remove:getUnused(${ref}, ${key})`);
      fres(supply.get(key));
    });
  }

  function supplyReset(key) {
    log(`supplyReset(${key})`);
    const plan = supply.get(key).plan;
    return new Promise((fres, frej) => cp.exec(
      `~/heroku addons:destroy ${key} -a ${app} --confirm ${app} >/dev/null && `+
      `~/heroku addons:create ${plan} --as ${key} -a ${app} >/dev/null && `+
      `~/heroku config -s -a ${app} | grep ^${key}`,
      (err, stdout) => {
        const r = stdout.toString();
        fres(supply.get(key).value = r.substring(r.indexOf('=')+2, r.length-2));
      }
    ));
  }

  function pendingRemove() {
    if(!unused.length || !pending.size) return;
    const ref = pending.keys().next().value;
    const fres = pending.get(ref);
    pending.delete(ref);
    const key = unused.shift();
    removed.set(ref, key);
    log(`pendingRemove:getUnused(${ref}, ${key})`);
    fres(supply.get(key));
    return ref;
  }

  function add(ref) {
    if(pending.has(ref)) {
      log(`add:removePending(${ref})`);
      pending.delete(ref);
    }
    if(removed.has(ref)) {
      const key = removed.get(ref);
      removed.delete(ref);
      log(`add:addUnused(${ref}, ${key})`);
      return supplyReset(key).then(() => {
        unused.push(key);
        pendingRemove();
        return ref;
      });
    }
    return Promise.resolve(ref);
  }

  return {add, remove, setup};
}
module.exports = HerokuAddonPool;
```