Import Notebook

Web-ui component

20201.10.24 김유정

1. 사용자가 입력한 URL을 파일을 가져올 수 있는 URL로 변환

2. session을 만들 수 있는 session launcher창 띄우기

3. 사용자가 입력한 정보를 바탕으로 session 만들기

사용자가 입력한 URL을 파일을 가져올 수 있는 URL로 변환

backend.ai-webui/src/components/backend-ai-import-view.ts

getNotebookFromURL()

↓

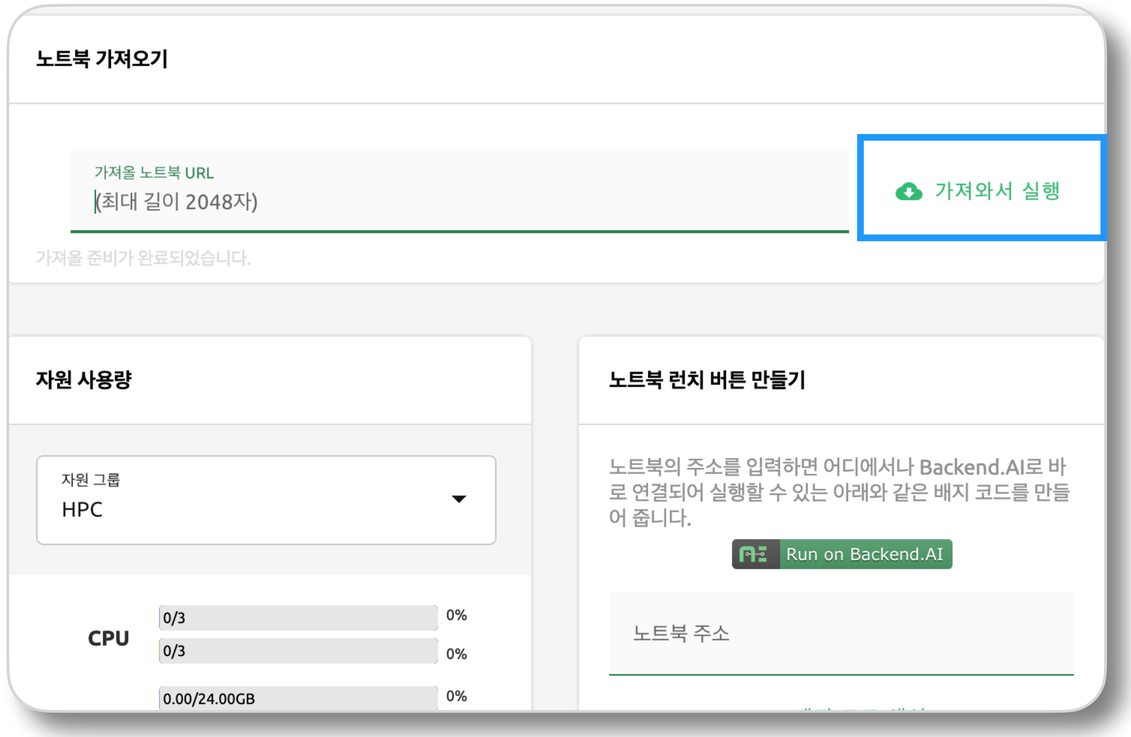
regularizeGithubURL(url)

↓

fetchNotebookURLResource(this.queryString)

↓

_fetchNotebookURLResource(downloadURL)



getNotebookFromURL()

사용자가 붙여넣은 주소 가져오기

regularizeGithubURL()

파일을 가져올 수 있는 URL로 변환

fetchNotebookURLResource(downloadURL)

변환된 URL을 다시 입력창에 붙여넣기

_fetchNotebookURLResource(downloadURL)

_launchSessionDialog() 함수 실행

```
getNotebookFromURL() {
  const url = this.shadowRoot.querySelector('#notebook-url').value;
  if (url !== '') {
    this.queryString = this.regularizeGithubURL(url);
    this.fetchNotebookURLResource(this.queryString);
  }
}

regularizeGithubURL(url) {
  url = url.replace('/blob/', '/');
  url = url.replace('github.com', 'raw.githubusercontent.com');
  return url;
}

fetchNotebookURLResource(downloadURL): void {
  this.shadowRoot.querySelector('#notebook-url').value = downloadURL;
  if (typeof globalThis.backendaiclient === 'undefined' || globalThis.backendaiclient === null
    document.addEventListener('backend-ai-connected', () => {
      this._fetchNotebookURLResource(downloadURL);
    }, true);
  } else { // already connected
    this._fetchNotebookURLResource(downloadURL);
  }
}

_fetchNotebookURLResource(downloadURL) {
  fetch(downloadURL).then((res) => {
    this.notification.text = _text('import.ReadyToImport');
    this.importMessage = this.notification.text;
    this.notification.show();
    this.sessionLauncher.selectDefaultLanguage(true, this.environment);
    this.sessionLauncher.importScript = '#!/bin/sh\ncurl -O ' + downloadURL;
    this.sessionLauncher.importFilename = downloadURL.split('/').pop();
    this.sessionLauncher._launchSessionDialog();
  }).catch((err) => {
    this.notification.text = _text('import.NoSuitableResourceFoundOnGivenURL');
    this.importMessage = this.notification.text;
    this.notification.show();
  });
}
```

session을 만들 수 있는 session launcher창 띄우기

backend.ai-webui/src/components/backend-ai-session-launcher.ts

_launchSessionDialog()

↓

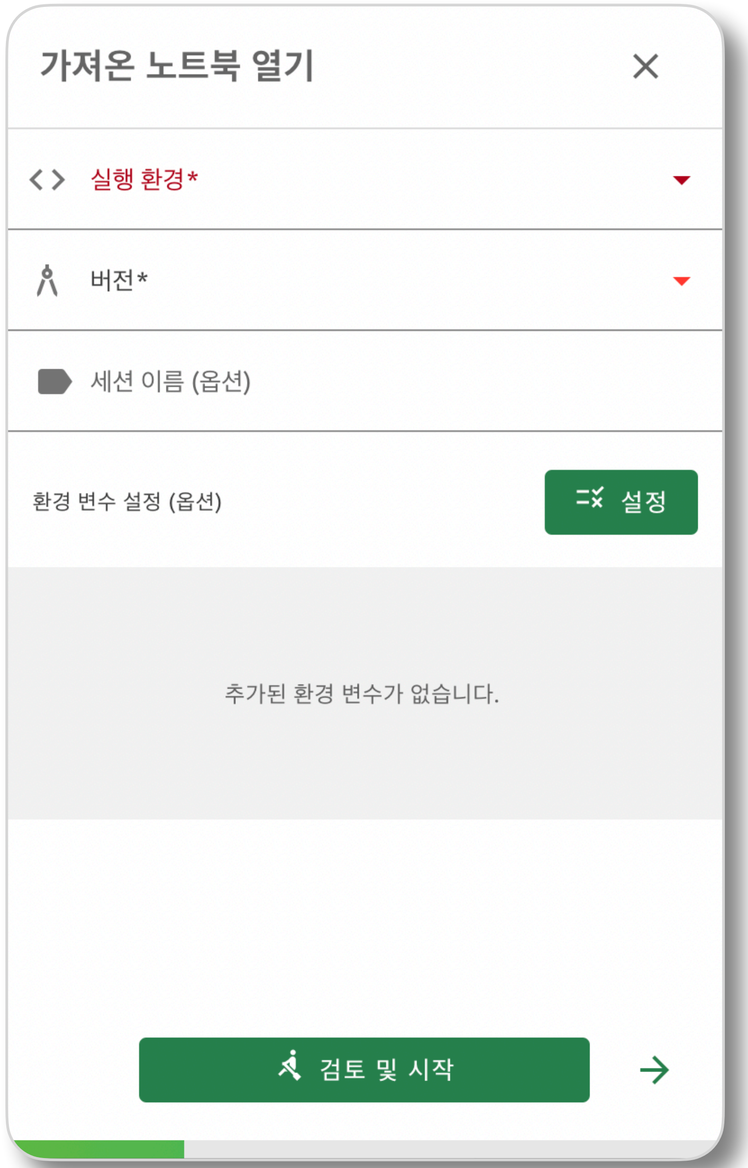
_resetProgress()

↓

_updateSelectedScalingGroup()

↓

requestUpdate()



_launchSessionDialog()

세션런처창의 정보들을 초기화하고 requestUpdate()로 업데이트 한 후 세션런처창을 띄움(=new-session-dialog 아이디를 가진 요소를 보여줌)

_resetProgress()

Progress 정보 초기화. Session launcher 의 첫번 째 페이지로 이동

_updateSelectedScalingGroup()

자원그룹 초기화

```
/**
 * If backendaiclient is not ready, notify wait for initializing.
 * Else, launch session dialog.
 * */
async _launchSessionDialog() {
  if (typeof globalThis.backendaiclient === 'undefined' || globalThis.backendaiclient === null || g
    setTimeout(() => {
      this._launchSessionDialog();
    }, 1000);
    // this.notification.text = _text('session.launcher.PleaseWaitInitializing');
    // this.notification.show();
  } else {
    this.folderMapping = Object();
    this._resetProgress();
    await this.selectDefaultLanguage();
    // Set display property of ownership panel.
    const ownershipPanel = this.shadowRoot.querySelector('wl-expansion[name="ownership"]');
    if (globalThis.backendaiclient.is_admin) {
      ownershipPanel.style.display = 'block';
    } else {
      ownershipPanel.style.display = 'none';
    }
    this._updateSelectedScalingGroup();
    /* To reflect current resource policy */
    await this._refreshResourcePolicy();
    this.requestUpdate();
    this.shadowRoot.querySelector('#new-session-dialog').show();
  }
}
```

```
/**
 * Move to first page and initialize environment variables and selected mount folders.
 *
 */
_resetProgress() {
  this.moveProgress(-this.currentIndex + 1);
  this._resetEnvironmentVariables();
  this._unselectAllSelectedFolder();
}
```

사용자가 입력한 정보를 바탕으로 session 만들기

backend.ai-webui/src/components/backend-ai-session-launcher.ts
backend.ai-webui/src/lib/backend.ai-client-node.ts

_newSession()

↓

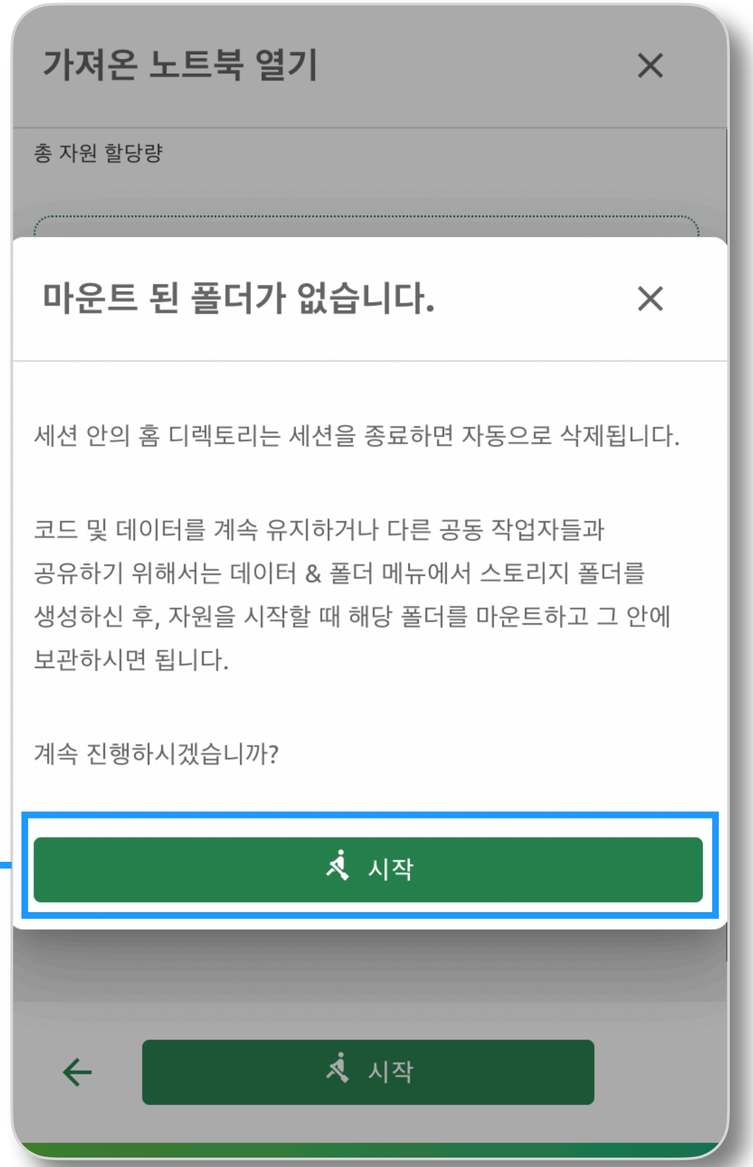
_createKernel()

↓

createIfNotExists()

↓

“POST” REST API 호출



_newSession()

config라는 객체를 만들어 client에 대한 정보를 넣어줌
sessions라는 배열에 kernelName, sessionName, config 추가
_createKernel함수 실행

```
> sessions
< ▼ [{…}] i
    ▼ 0:
      ▶ config: {group_name: 'gardener', domain: 'default', scaling_group: undefined, cluster_mode: u…
        kernelName: "kernelName"
        sessionName: "sessionName"
      ▶ [[Prototype]]: Object
        length: 1
    ▶ [[Prototype]]: Array(0)
```

```
/**
 * Make a new session.
 * */
_newSession() {
  const confirmationDialog = this.shadowRoot.querySelector('#launch-confirmation-dialog');
  confirmationDialog.hide();
  let kernel;
  let version;

 ...
this.scaling_group = this.shadowRoot.querySelector('#scaling-groups').value;
const config = {};
config['group_name'] = globalThis.backendaiclient.current_group;
config['domain'] = globalThis.backendaiclient._config.domainName;
config['scaling_group'] = this.scaling_group;
if (globalThis.backendaiclient.supports('multi-container')) {
  config['cluster_mode'] = this.cluster_mode;
  config['cluster_size'] = this.cluster_size;
}
config['maxWaitSeconds'] = 15;
const ownerEnabled = this.shadowRoot.querySelector('#owner-enabled');
if (ownerEnabled && ownerEnabled.checked) {
  config['group_name'] = this.shadowRoot.querySelector('#owner-group').value;
  config['domain'] = this.ownerDomain;
  config['scaling_group'] = this.shadowRoot.querySelector('#owner-scaling-group').value;
  config['owner_access_key'] = this.shadowRoot.querySelector('#owner-accesskey').value;
  if (!config['group_name'] || !config['domain'] || !config['scaling_group'] || !config ['owner_access_key']) {
    this.notification.text = _text('session.launcher.NotEnoughOwnershipInfo');
    this.notification.show();
    return;
  }
}

 ...
 if (this.num_sessions > 1) {
   for (let i = 1; i <= this.num_sessions; i++) {
     const add_session = {'kernelName': kernelName, 'sessionName': `${sessionName}-${randStr}-${i}`, config};
     sessions.push(add_session);
   }
 } else {
   sessions.push({'kernelName': kernelName, 'sessionName': sessionName, config});
 }
 const createSessionQueue = sessions.map((item) => {
   return this.tasker.add('Creating ' + item.sessionName, this._createKernel(item.kernelName, item.sessionName, item.config), '', 'session');
 });
 Promise.all(createSessionQueue).then((res: any) => {…
 }).catch((err) => {…
 });
```

_createKernel

backend.ai-webui/src/components/backend-ai-session-launcher.ts

backend.ai-client-node.ts의 createIfNotExists 함수 호출

```
_createKernel(kernelName, sessionName, config) {
  const task = globalThis.backendaiclient.createIfNotExists(kernelName, sessionName, config, 20000);
  task.catch((err) => {…
  });
  return task;
}
```

createIfNotExists()

backend.ai-webui/src/lib/backend.ai-client-node.ts

newSignedRequest()함수를 통해
_wrapWithPromise에서 사용할 URL 생성

```
async createIfNotExists(kernelType, sessionId, resources = {}, timeout: number = 0) {
  if (typeof sessionId === 'undefined' || sessionId === null)
    sessionId = this.generateSessionId();
  let params = {
    "lang": kernelType,
    "clientSessionToken": sessionId,
  };
  if (resources != {}) {…
  }
  let rqst;
  if (this._apiVersionMajor < 5) { // For V3/V4 API compatibility
    rqst = this.newSignedRequest('POST', `${this.kernelPrefix}/create`, params);
  } else {
    rqst = this.newSignedRequest('POST', `${this.kernelPrefix}`, params);
  }
  //return this._wrapWithPromise(rqst);
  return this._wrapWithPromise(rqst, false, null, timeout);
}
```

_wrapWithPromise()

backend.ai-webui/src/lib/backend.ai-client-node.ts

Request 정보 설정 후 fetch()함수를 통해서
Backend.ai manager의 “POST” REST API를 호출

```
async _wrapWithPromise(rqst, rawFile = false, signal = null, timeout: number = 0, retry: number = 0) {
  let errorType = Client.ERR_REQUEST;
  let errorTitle = '';
  let errorMsg;
  let errorDesc = '';
  let resp, body, requestTimer;

  resp = await fetch(rqst.uri, rqst);
  if (typeof requestTimer !== "undefined") {
    clearTimeout(requestTimer);
  }
```

```python
def create_app(default_cors_options: CORSOptions) -> Tuple[web.Application, Iterable[WebMiddleware]]:
    app = web.Application()
    app.on_startup.append(init)
    app.on_shutdown.append(shutdown)
    app['api_versions'] = (1, 2, 3, 4)
    app['session.context'] = PrivateContext()
    cors = aiohttp_cors.setup(app, defaults=default_cors_options)
    cors.add(app.router.add_route('POST', '', create_from_params))
    cors.add(app.router.add_route('POST', '/_/create', create_from_params))
    cors.add(app.router.add_route('POST', '/_/create-from-template', create_from_template))
    cors.add(app.router.add_route('POST', '/_/create-cluster', create_cluster))
    cors.add(app.router.add_route('GET',  '/_/match', match_sessions))
    session_resource = cors.add(app.router.add_resource(r'/{session_name}'))
    cors.add(session_resource.add_route('GET',    get_info))
    cors.add(session_resource.add_route('PATCH',  restart))
    cors.add(session_resource.add_route('DELETE', destroy))
    cors.add(session_resource.add_route('POST',   execute))
    task_log_resource = cors.add(app.router.add_resource(r'/_/logs'))
    cors.add(task_log_resource.add_route('HEAD', get_task_logs))
    cors.add(task_log_resource.add_route('GET',  get_task_logs))
    cors.add(app.router.add_route('GET',  '/{session_name}/logs', get_container_logs))
    cors.add(app.router.add_route('POST', '/{session_name}/interrupt', interrupt))
    cors.add(app.router.add_route('POST', '/{session_name}/complete', complete))
    cors.add(app.router.add_route('POST', '/{session_name}/shutdown-service', shutdown_service))
    cors.add(app.router.add_route('POST', '/{session_name}/upload', upload_files))
    cors.add(app.router.add_route('GET',  '/{session_name}/download', download_files))
    cors.add(app.router.add_route('GET',  '/{session_name}/download_single', download_single))
    cors.add(app.router.add_route('GET',  '/{session_name}/files', list_files))
    return app, []
```

Thank you :)